Arlang : langage de formulation et de programmation à l'usage des architectes

Jacques Autran, Marius Frégier, Miguel Rodriguez, Jacques Zoller, Eugène Chouraqui, Philippe Roussel

gamsau

groupe d'études pour l'application des methodes scientifiques
à l'architecture et l'urbanisme
unité pedagogique Marseille Luminy

arlang

langage de formulation et de programmation à l'usage des architectes

J.AUTRAN M.FREGIER M.RODRIGUEZ J.ZOLLER

PE D'ÉTUDES POUR L'APPLICATION DES MÉTHODES SCIENTIFIQUES A L'ARCHITECTURE ET L'URBANISME
D'ARCHITECTURE DE MARSEILLE-LUMINY - 13 - MARSEILLE (9e) - TELEPHONE (91) 41.06.71 ET 41.17.02

ARLANG : langage de programmation à l'usage des architectes

La présente recherche a pu être menée au sein du GAMSAU grâce au concours du Comité pour la Recherche et le Développement en Architecture [DGRST - CORDA] de Janvier 1975 à Mai 1975 - Contrat n° 75.73.002.00.202.75.01, sur le thème "Langage architectural de programmation".

Nous remercions Eugène CHOURAQUI et Philippe ROUSSEL qui, par leur contribution technique et critique, ont guidé notre recherche.

Cette édition reproduit le contenu du rapport de fin de contrat.

Mai 1976

Etude réalisée par :

Jacques AUTRAN	Architecte
Marius FREGIER	Architecte
Miguel RODRIGUEZ	Informaticien
Jacques ZOLLER	Informaticien

avec la collaboration de :

Eugène CHOURAQUI	Laboratoire d'Informatique pour les Sciences de l'Homme, CNRS Marseille
Philippe ROUSSEL	Groupe Intelligence Artificielle de Marseille Luminy

CORDA 1976

La présente recherche a pu être menée au sein du GAMSAU grâce au concours du Comité pour la Recherche et le Développement en Architecture (DGRST - CORDA) de Janvier 1975 à Mai 1976 - Contrat n° 75.73.002.00.202.75.01, sur le thème "Langage architectural de programmation".

Nous remercions Eugène CHOURAQUI et Philippe ROUSSEL qui, par leur contribution technique et critique, ont guidé notre recherche.

Cette édition reproduit le contenu du rapport de fin de contrat.

Mai 1976

SOMMAIRE

3 - Restitution des informations

31 Les fonctions consultatives

311 Consultation des champs

3111 Valeur d'un noeud
3112 Descripteur associé à un noeud
3113 Relations entre deux noeuds
3114 Fonctions de listage associées à un noeud
3115 Ancêtres d'un élément de base
3116 Niveau d'un noeud

312 Consultation de descripteurs

3121 Eléments d'un descripteur
3122 Relations de descripteurs élémentaires dans un descripteur
3123 Opérateurs dans un descripteur
3124 Familles d'un descripteur

313 Consultation du vocabulaire

3131 Familles d'un élément du vocabulaire
3132 Eléments d'une famille

32 Instruction de sortie

4 - Opérations de traitement

41 Expressions et définition d'opérateurs et de comparateurs

411 Les expressions

4111 Expressions numériques
4112 Expressions logiques
4113 Expressions chaîne
4114 Expressions descripteur
4115 Expressions accès

412 Définition d'opérateurs et de comparateurs

4121 Définition d'opérateurs
4122 Définition de comparateurs

PARTIE I : Architecture et informatique

Nous présentons dans cette partie une brève analyse de l'état actuel des recherches dans le domaine de l'utilisation de l'informatique en architecture.

PARTIE I : Architecture et informatique

Nous présentons dans cette partie une brève analyse de l'état actuel des recherches dans le domaine de l'utilisation de l'informatique en architecture.

1 - INFORMATIQUE ET ARCHITECTURE : BILAN CRITIQUE

Le GAMSAU, groupe de recherche et groupe de 3me cycle de l'Unité Pédagogique d'Architecture de Marseille-Luminy, a effectué depuis sa création (1970) un ensemble de travaux concernant diverses préoccupations des intervenants dans les disciplines de l'architecture et de l'urbanisme (1).

L'idée de la présente recherche est issue de réflexions sur les problèmes rencontrés en cours de réalisation ou à la suite de ces travaux. Elle est en concordance avec l'évolution actuelle des recherches en informatique appliquée à l'architecture et doit donc être resituée dans le contexte de l'évolution générale des recherches menées en ce domaine (2). On peut caractériser cette évolution par une première phase d'enthousiasme et cours de laquelle les équipes de recherches, confiantes en la puissance évidente de l'ordinateur (capacités de mémorisation, rapidité de calcul), ont cru libérer le concepteur des activités de projet distrayantes et non productives. Les programmes informatiques réalisés furent d'abord indépendants, puis reliés entre eux pour couvrir l'ensemble du processus de conception; utilisés indépendamment ils se révélèrent pour la plupart inopérants du fait de la disproportion entre la pauvreté des données formalisées et celle des résultats obtenus et la richesse des objets architecturaux. C'était méconnaître la complexité spécifique des problèmes architecturaux, difficilement formalisables. D'autre part, chaque programme ne traitant que de problèmes bien spécifiques, leur puissance de traitement ne pouvait compenser les limites trop étroites de leur domaine d'application. A cette carence d'ordre sémantique due à l'inadéquation des langages de programmation utilisés pour l'écriture des programmes, s'ajoutaient la difficulté et la durée du codage des données d'entrée à la fois complexe et différent pour chacun d'eux.

L'enchaînement des programmes écrits indépendamment fut une solution apportée à ce dernier inconvénient, supprimant ainsi les phases de formalisation intermédiaires : on entre les données de base au début de la chaîne, il en sort un résultat en bout de chaîne. Outre les problèmes techniques d'interfaces qui ont pu se poser, obligeant le plus souvent à réécrire les programmes, des problèmes d'ordre méthodologique apparurent : l'enchaînement des programmes de conception devait sous-tendre un processus de conception nécessairement déterministe.

Un deuxième échec quant à l'utilisation de ces chaînes est dû à la méconnaissance d'un caractère essentiel du processus de conception : son aspect heuristique. Grâce à la disponibilité, sur le marché, de terminaux avec lesquels l'interaction entre l'homme et la machine devenait plus aisée, apparaissait la notion d'aide à la conception. Grâce à son caractère interactif, appuyé sur un matériel adapté aux manipulations spécifiques de la pratique architecturale (console de visualisation graphique avec light pen, tablette graphique, etc ...), un système de conception assistée permettrait à l'architecte de se servir lui-même de la machine : pouvoir gérer à sa manière le processus de conception,

(1) Rapport d'activités et de recherches 1970-1976, Cahier n°21, Juin 1976

(2) Computer aids to Design and Architecture, edited by Nicholas NEGROPONTE

et augmenter son pouvoir de prise de décision, dans les phases faisant intervenir l'intuition du créateur.

L'emploi de terminaux sophistiqués permettrait du même coup la réduction des difficultés de codage des données d'entrée et de lecture des résultats de traitement. URBAN V (1) est une expérience qui, bien qu'ayant échoué par méconnaissance de techniques de programmation, actuellement résolues, a eu le mérite de soulever un ensemble de problèmes relatifs à la conception assistée : le système de représentation des objets, trop simplifié pour envisager une application opérationnelle, permettait cependant de mettre en place un système de traitement rendant d'importants services à l'expérimentation du premier système d'interaction homme-machine en conception architecturale. Les problème de fond, bien que soulevés, ne furent pas résolus. D'autres systèmes interactifs furent mis en place, mais la plupart échouèrent; les expériences effectuée en Grande Bretagne sont les seules qui, à notre connaissance, sont devenues opérationnelles, car opérant dans un monde architectural limité aux domaines formalisables.

Les deux obstacles essentiels à un succès des systèmes interactifs de conception assistée en architecture sont à notre avis :

1. Une connaissance insuffisante des procédures de décision en conception architecturale,

2. L'inadéquation des techniques et des langages de programmation à la description et à la manipulation des objets.

La nécessité d'utiliser l'informatique dans le domaine de l'architecture est aujourd'hui remise en cause par de nombreux architectes : les échecs des diverses tentatives présentées ont découragé certains, et confirmé le scepticisme d'autres. On oublie trop souvent la relative jeunesse de cet aspect de l'architecture; sa complexité de plus en plus ressentie lui donne la plupart du temps comme référence le domaine de la recherche plutôt que celui de la pratique opérationnelle.

Les capacités de traitement par ordinateur de problèmes de conception architecturale sont cependant indiscutables : il reste à déterminer quelles sont les procédures informalisables dans le processus de conception. Nous pensons que ce travail est de longue haleine, et que par conséquent les procédures informatisables doivent être introduites dans le secteur opérationnel au fur et à mesure de leur mise au point. Nous ne connaissons actuellement aucun outil qui permette la réalisation de cet objectif dans le contexte spécifique de l'architecture : c'est l'élaboration d'un tel outil que nous proposons.

Les progrès effectués en informatique depuis quelques années permettaient d'envisager un nouvel essor de la conception assistée. Les connaissances méthodologiques n'ont été approfondies que depui peu, et sont encore au stade de l'expérimentation. L'idée de base de notre recherche fut donc qu'un langage de programmation à l'usage des architectes permettant la description et la manipulation d'objets architecturaux dans le contexte de la conception architecturale était une voie à suivre.

(1) Nicholas NEGROPONTE - MIT Cambridge Massachusetts

2 - OBJECTIFS ET MOYENS POUR LA MISE EN OEUVRE DE L'OUTIL PROPOSE

Dans l'optique de notre recherche dont le but était donc d'élaborer un langage de programmation adapté aux besoins de l'architecte, il convenait de prendre en compte les résultats des expériences passées afin de retenir leurs aspects positifs et d'éliminer dans les solutions apportées au terme de la présente recherche les caractères qui les avaient rendues inopérantes.

En réponse aux objectifs présentés dans ce paragraphe, la partie 2 du présent document est un exposé de l'outil proposé : "ARLANG".

Le premier objectif, déterminé dès le début de notre recherche, fut que le langage que nous devions élaborer ne devait pas décrire les objets manipulés par l'architecte, mais permettre la description et la manipulation de ces objets. Cette ambiguité est très souvent apparue chez les personnes auxquelles nous avons exposé le langage, c'est pourquoi il convient de préciser cet objectif dès le départ : de la même manière qu'EUCLID (1) est un outil permettant de créer et de manipuler toute forme géométrique, ARLANG permettra la description et la manipulation de tout objet selon des procédures écrites par l'architecte (nous ne réalisons pas la description des objets architecturaux, ni des programmes de manipulation de ces objets). Les modes de description d'objets et les outils de manipulation ne pourront être élaborés qu'au fur et à mesure de l'utilisation d'un tel langage, moyennant un progrès dans la connaissance des informations architecturales (données et procédures).

Formalisation des données :

Les travaux antérieurs ont montré :

- La multiplicité des façons de décrire des objets pour leur traitement par des programmes dont la tâche est identique,
- L'aspect opératoire des descriptions, liées au programme de traitement,
- La complexité du codage desdescriptions, trop proche de la machine,
- La longueur du codage des données pour certains programmes (perspectives automatiques par exemple), dans la mesure où ce codage ne sert qu'à un programme effectuant une tâche très partielle.

Afin de supprimer ces inconvénients, notre objectif fut :

- d'unifier toutes les descriptions d'objets en fournissant un outil permettant de décrire tout objet architectural selon une même structure,
- que l'outil de description proposé ne soit pas lié à des présupposés opératoires, mais permette une description naturelle de l'objet,
- que le langage de description soit le plus proche possible du langage de l'architecte, de façon à être aisément compréhensible et utilisable.
- à partir du moment où l'on proposait un outil permettant de décrire des objets sans présupposer des manipulations à effectuer sur ceux-ci, on pouvait envisager que des programmes divers y soient appliqués. On résolvait ainsi en partie le problème de longueur du codage.

(1) EUCLID - BRUN et THERON - LIMSI. CNRS

Programmation:

La situation actuelle qui se caractérise par le spécialiste-informaticien écrivant des programmes d'aide à la conception pour l'architecte tend à éloigner (ou à maintenir éloigné) l'architecte de l'informatique : le phénomène "boite noire" peut rebuter nombre de concepteurs soucieux de connaître les procédures qui l'amèneront à une prise de décision consciente. De plus, les programmes de traitement restent épars et spécifiques à un type de machine, de langage, ...

C'est pourquoi nous proposerons un langage de programmation qui soit non seulement langage descriptif mais aussi langage algorithmique, c'est-à-dire intégrant les instructions habituelles de ce type de langage, complétées de procédures plus sophistiquées (récursivité, non-déterminisme, etc ...). Nous pensons alors que le concepteur-architecte pourra ainsi créer lui-même, et au fur et à mesure de ses besoins, ses propres programmes de traitement. Le spécialiste-informaticien aurait alors pour rôle la résolution de problèmes complexes de programmation. On devra également rendre possible la conservation en bibliothèque de certains de ces programmes, pour permettre leur réutilisation dans des procédures analogues, par des concepteurs qui pourront être différents de celui qui a écrit le programme.

Apprentissage de la programmation:

Proposer un outil de programmation semble un objectif incompatible avec le fait que cet outil s'adresse à des non-informaticiens. Nous avons donc essayé de fournir le maximum de facilités, au travers des techniques interactives et conversationnelles pour aider cette programmation. D'autre part, le langage est conçu de manière à se rapprocher le plus possible du langage parlé (description ou manipulation). Une formation de base minimum semble cependant indispensable pour l'enseignement de méthodes de programmation.

Objectifs méthodologiques:

Les procédures utilisées par l'architecte sont itératives et nécessitent son intervention au fur et à mesure de l'évolution de son travail, il y a création d'informations et par ce fait enrichissement de la connaissance qu'il a sur le projet qu'il étudie. Les possibilités de résolution d'un problème ne peuvent pas être déterminées à l'avance, mais seulement au fur et à mesure que l'architecte acquiert de l'information sur l'objet qu'il traite.

Le choix des procédures méthodologiques utilisées est essentiellement empirique et dépend de l'architecte et du projet sur lequel il travaille. Il convient donc de laisser la liberté au concepteur de choisir une méthode de travail adaptée.

Pour permettre cela, deux principes méthodologiques devaient être énoncés :

1/ Permettre la conservation de l'information (données et procédures de traitement) et de son évolution en cours de conception,

2/ Permettre la modification de l'information (feed-back) en cours de conception et à tout moment du processus.

PARTIE II : Description du langage ARLANG

L'ensemble des objectifs généraux définis en début de recherche nous ont permis de guider et de contrôler l'élaboration du langage et de son environnement. C'est une réponse à ces objectifs que nous proposons dans cette deuxième partie.

CHAPITRE 1 : LES FONDEMENTS DU LANGAGE ARLANG

CHAPITRE I : LES FONDEMENTS DU LANGAGE ARLANG

1 - Rappel des objectifs de la recherche

Depuis une décennie on a assisté à l'éclosion de nombreux langages évolués de programmation, chacun apportant de nouveaux concepts ou de nouvelles techniques plus ou moins sophistiqués, mais tous restant définis sur un même plan purement opératoire, donc de façon très liée à la machine.

La tendance récente veut au contraire mettre l'accent sur l'aspect "dénotationnel" des langages, c'est-à-dire de définir les formalismes incluant bien sûr toutes les techniques évoluées, mais aussi et surtout de permettre à l'utilisateur de modéliser son problème.

Tout nouveau langage de programmation, qu'il soit orienté vers un type d'application précis (et c'est le cas d'ARLANG) ou non, doit donc être conçu de telle sorte que la programmation d'un problème et le choix de la représentation des êtres ou concepts manipulés, soit aussi proche que possible de la description de ce problème. Le langage devient alors à la fois un langage de description et d'assertion et un langage d'opération.

Cette approche a pour effets :

- de rendre plus transparente la pauvreté conceptuelle de la machine,
- d'aider l'utilisateur à structurer le problème, à traiter les données à manipuler,
- de rendre beaucoup plus lisible l'écriture des programmes,
- dans le cadre d'un langage orienté vers une discipline particulière, de faciliter la formalisation des concepts de base.

Bien entendu, dès qu'il s'agit de concevoir des langages de conception (et donc d'assertion) l'immense travail des logiciens modernes apporte une aide certaine, en particulier les limites immanquablement présentes dans tout nouveau formalisme (paradoxes, etc ...) peuvent être prédites et cernées et éviter ainsi au concepteur de croire en l'universalité de son produit.

ARLANG se situe en droite ligne dans cette approche de la programmation puisqu'il se veut à la fois :

- langage de programmation,
- langage de description des concepts (en utilisant un univers bien défini comme univers d'interprétation)
- langage d'assertion (relations entre les concepts).

Ainsi, un tel langage devra permettre :

- la réduction et la sélection de l'information pertinente
- la rationalisation du processus de conception
- l'augmentation des facultés de traitement de l'architecte
- la réduction de la durée d'étude d'un projet
- la prise en compte de programmes informatiques existants
- la transparence de l'aspect informatique
- la structuration pertinente des données d'un problème
- la communication aisée d'informations entre intervenants
- l'enrichissement de la connaissance architecturale

Ce rapport se veut l'exposé d'une réponse précise à l'ensemble de ces objectifs.

2 - ORGANISATION GENERALE DU LANGAGE

Afin de répondre aux objectifs présentés en 1, ARLANG a été conçu selon 3 volets :

1) Une structuration homogène de données exposée dans le § 3 (méta-langage).

2) L'élaboration d'un langage de programmation, qui, tout en offrant les outils classiques des autres langages actuels (FORTRAN, ALGOL, ...), prend en compte de plus la manipulation de la structure de données proposée et des techniques de programmation plus sophistiquées telles que le "non-déterminisme"

3) Une organisation générale du langage, permettant d'assurer avec facilité le transfert et le traitement de données architecturales, une gestion aisée des fichiers de programmes ou de données dans le cadre d'une mise en commun et d'une évolution de ces fichiers, et un apprentissage du langage et de la programmation relativement souple. Nous proposons de plus les types de configurations-ordinateur nécessaires à l'utilisation du langage.

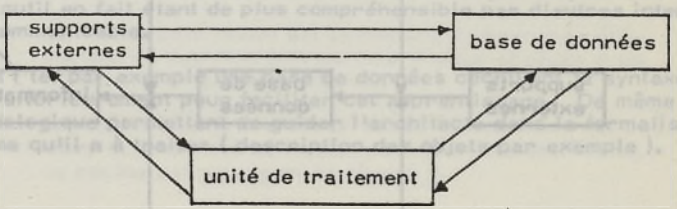
Ce sont ces 3 principaux points qui seront l'objet de ce paragraphe.

21 - Les grandes fonctions d'ARLANG :

Comme tout langage informatique, ARLANG est un outil de transfert d'information et de traitement.

Cependant, en tant que langage non seulement algorithmique, mais aussi descriptif, il propose l'enregistrement des données architecturales au sein d'une base de données dont la méthode de structuration est prédéfinie.

L'existence de cette base de données, aux côtés de l'unité de traitement et de supports externes, nous permet de proposer le schéma suivant, indiquant les divers types d'opérations possibles en ARLANG pour les échanges et le traitement d'informations.



- Si l'on se place du point de vue de la base de données, ARLANG propose des outils

  - d'enregistrement d'informations (en provenance de l'extérieur ou de l'unité de traitement) dans la base de données
  - de restitution d'informations contenues dans la base de données à fin de traitement immédiat ou ultérieur manuel ou automatique

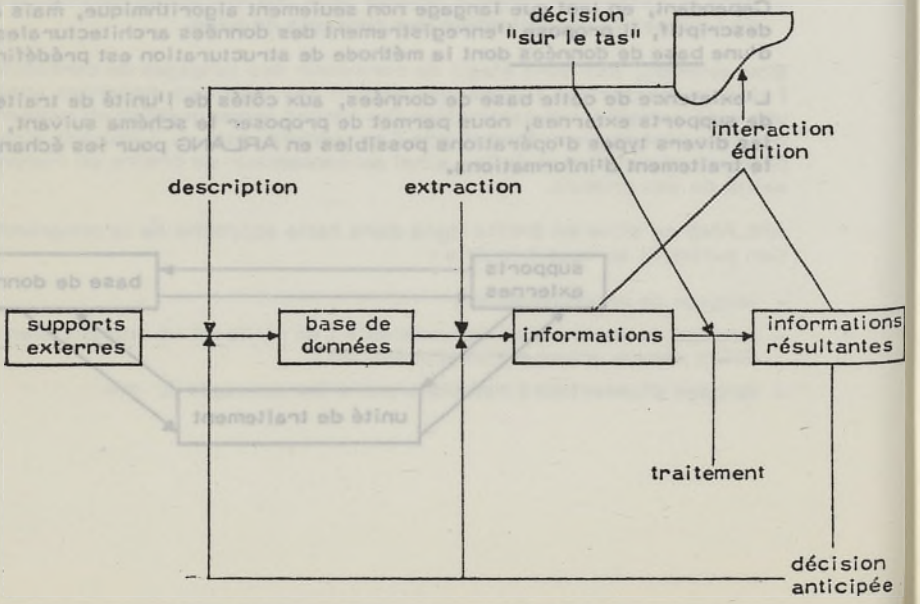
- Si l'on se place du point de vue de l'extérieur, on peut :

  - recevoir des informations en provenance de la base de données ou de l'unité de traitement
  - enregistrer des informations dans la base de données ou dans l'unité de traitement pour traitement automatique immédiat ou ultérieur

- Si l'on se place du point de vue de l'unité de traitement, on peut :

  - lui envoyer des informations en provenance de la base de données ou de l'extérieur
  - transférer des informations de l'unité de traitement vers l'extérieur ou la base de données
  - traiter des données de l'unité de traitement afin d'en créer d'autres ou de faire un choix.

Un exemple schématique de processus de programme ARLANG sera :



22 - Bases de données et bases de programmes

Afin de rendre ARLANG ouvert et évolutif, une gestion automatique des fichiers de données et des fichiers de programme est mise en place, répondant aux critères suivants :

- possibilité de création de bases de données et de bases de programmes personnelles communes ou publiques

- possibilité de mémorisation de ces bases, de leur réutilisation et de leur enrichissement

- possibilité de communication et de protection de ces bases

Notons qu'en ARLANG le programmeur n'aura pas à manipuler de cartes de contrôle de gestion des fichiers. L'accès aux fichiers et leur protection sera assuré par l'emploi de "clés d'accès" (cf. CH.2 - 1121 pour les bases de données, CH.2-1123 pour les bases de programme).

23 - Apprentissage du langage et de la programmation

Afin de rendre l'approche du langage ARLANG plus aisée, nous avons introduit au niveau de la programmation (écriture de programme) les outils classiques que l'on trouve dans la plupart des langages conversationnels : analyse syntaxique immédiate de chaque entité écrite et possibilité, grâce à un éditeur de texte, de modifier une instruction.

Ces outils doivent permettre un apprentissage plus rapide de la syntaxe, ainsi qu'une modification aisée des instructions d'un programme.

Moins classique est la possibilité offerte dans le langage de puiser, en cours de programmation, des informations dans une base de données, l'examen de ces données permettant la continuation de l'écriture du programme ("feed-forward"). Ceci est assuré grâce aux deux modes possibles d'un programme ARLANG (mode exécutable et mode bureau) et à la possibilité d'intégrer en cours d'écriture d'un programme en mode exécutable une session d'instructions (en mode bureau) immédiatement exécutées.

De plus, nous avons pensé que munir ARLANG d'une structure unique de données permettrait à l'architecte de mieux structurer ses informations, la description qu'il en fait étant de plus compréhensible par d'autres intervenants, donc communicable.

D'autres outils (tel par exemple une base de données décrivant la syntaxe) seront fournis ultérieurement pour faciliter cet apprentissage. De même un outil méthodologique permettant de guider l'architecte dans la formalisation du problème qu'il a à traiter (description des objets par exemple).

24 - Configurations du matériel

Pour permettre une utilisation optimale d'ARLANG, la configuration minimale du matériel-ordinateur doit être constituée comme suit :

- un mini-ordinateur de 32 K
- un disque
- une console
- un lecteur de cartes
- une imprimante 300 lignes/minute
- un écran graphique

Ce matériel étant muni de plus d'un compilateur FØRTRAN interactif.

3 - LE METALANGAGE DE DESCRIPTION

Bien qu'au sens habituel en informatique, un métalangage soit un moyen symbolique de représenter les équations définissant un langage de programmation, nous avons utilisé le terme de métalangage ARLANG dans le sens de formalisation d'objets architecturaux.

31 - Description structurée des données en ARLANG

La formalisation des données en ARLANG est faite grâce à un support d'information présentant une structure d'arbre, enrichie grâce à l'introduction d' "opérateurs" logiques et grâce à la possibilité de spécifier des relations entre éléments de l'arbre.

Il convient, avant d'expliciter les éléments du métalangage de description, que nous donnions la définition des termes relatifs à la structure en arbre adoptée que nous utilisons.

311 - Définitions et notations

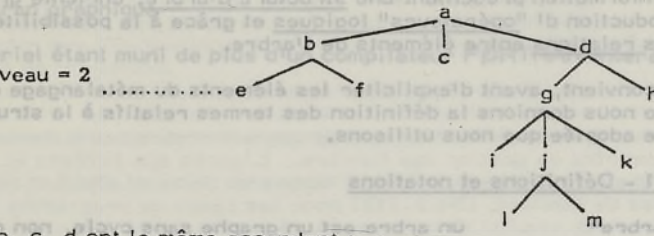
"Arbre"	un arbre est un graphe sans cycle, non orienté et connexe
"Noeud"	un noeud d'un arbre est un sommet de l'arbre auquel est associé le nom ou la qualité d'un objet
"Branche"	une branche d'un arbre établit la correspondance entre 2 noeuds d'un arbre; une branche est soit élémentaire (segment joignant 2 noeuds contigus d'un arbre), soit composée de plusieurs branches élémentaires.
"Chemin"	un chemin d'un noeud à un autre de niveau supérieur (qui a la propriété d'être "descendant" de ce noeud) est constitué de l'ensemble des branches élémentaires permettant de partir de ce noeud et d'atteindre l'autre.
"Descendants"	un noeud possède des descendants s'il existe un ou plusieurs chemin(s) partant de ce noeud et permettant d'atteindre le ou les descendants.
"Ascendants"	un noeud possède des ascendants si ce noeud est descendant d'autres noeuds (qui sont ses ascendants).
"Niveau"	le niveau d'un noeud est le nombre de branches élémentaires que l'on doit parcourir pour accéder à ce noeud en partant de la racine.
"Racine"	la racine d'un arbre est un noeud particulier, qui possède la propriété de ne pas posséder d'antécédents; la racine est unique et de niveau 0.

"Sous-arbre" un arbre B est un sous-arbre de l'arbre A au noeud a si et seulement si l'ensemble des noeuds de B est l'ensemble des descendants du noeud a dans A.

"Feuille" c'est un noeud qui possède la propriété de ne pas avoir de descendants.

Notation :

x notation schématique (cf. récapitulation en Annexe 1.1)



- b, c, d ont le même ascendant : a
- g a comme ascendants d et a
- j a comme descendants l et m
- a, b, c, ... m sont les noeuds de l'arbre
- d: g, d: g: k sont des exemples de branches de l'arbre
- d: g: j: m est le chemin de d à m
- i, j, k sont de même niveau = 3
- a est la racine de l'arbre
- les noeuds i, j, k, l, m sont des noeuds du sous-arbre de A au noeud g
- e, m sont deux exemples de feuilles

x notation fonctionnelle :

on peut passer de la notation schématique d'un arbre à une notation dite "fonctionnelle"; l'arbre schématisé ci-dessus sera ainsi noté :

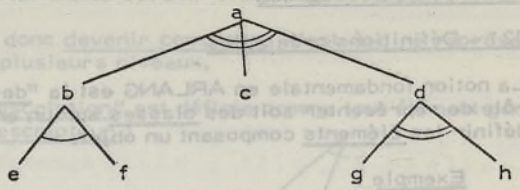
a (b (e, f), c, d (g (i, j (l, m), k), h))

312 - Enrichissement de la structure arborescente de base : introduction des opérateurs logiques et et ou

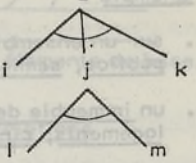
Afin de rendre la structure de base plus riche, nous avons introduit au niveau de la structure les deux opérateurs booléens ou et et, introduisant ainsi une différenciation sémantique au niveau des objets décrits.

L'affectation de l'opérateur sémantique ou ou et sera effectuée au moment de la définition des descendants de niveau n + 1 d'un noeud de niveau n.

Sur l'exemple précédent, on pourra écrire par exemple :

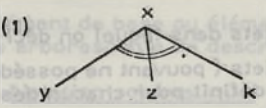


(3)



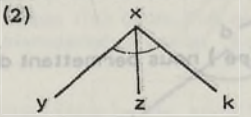
Notation schématique

(1)



signifie que l'on munit les branches de l'opérateur ou

(2)



signifie que l'on munit les branches de l'opérateur et

Notation fonctionnelle

(1) x ((y, z, k))
(2) x (y, z, k)
(3) a ((b (e, f), c, d ((g (i, j (e, m) , k), h))))

- le sens d'une telle arborescence munie des opérateurs ou ou et sera, pour les 3 cas de figure décrits :

(1) x : y ou x : z ou x : k
(2) x : y et x : z et x : k
(3) a : b : e et a ; b : f ou a : c ou a : d : g : i et a : d : g : j : l et a : d : g : j : m et a : d : g : k ou a : d : h

Ceci constitue les "interprétations sémantiques" de chacune des 3 arborescences. Nous verrons par la suite la signification du et et du ou dans un contexte architectural.

32 - Expressions de création des unités permettant d'effectuer une description en ARLANG

321 - Définitions préalables

La notion fondamentale en ARLANG est la "description". Elle a pour rôle de représenter soit des classes sur un ensemble d'objets, soit de définir les éléments composant un objet.

Exemple :

- sur un ensemble d'espaces, on pourra définir ceux qui sont publics, semi-publics ou privés
- un immeuble de logements pourra être composé des éléments logements, circulations, hall, conciergerie, entrepôts.

Nous appelerons :

- "élément de base" un ensemble d'objets dans lequel on décrit des classes
  ou un ensemble d'objets (pouvant ne posséder qu'un seul élément) dont on définit pour chacun des éléments ses composants

Dans l'exemple précédent, ESPACES et IMMEUBLE DE LOGEMENTS sont 2 éléments de base

- "descripteurs" c'est l'unité (sous arbre) nous permettant de décrire un élément de base

Ainsi dans l'exemple :

{publics, 1/2 publics, privés} définissant des classes, ...
{logements, circulations, hall, conciergerie, entrepôts} définissant des composants est un descripteur.

Un descripteur est donc - un ensemble d'éléments ("éléments de descripteur")
et - un opérateur permettant de connaître le statut de ces éléments par rapport à l'élément de base qu'ils décrivent (propriétés définissant des classes ou éléments de l'objet).

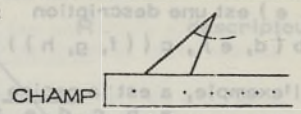
- "élément générateur" : c'est un élément de base décrit par un descripteur

- "une description" sera donc l'association d'un élément générateur et d'un descripteur

- Il convient de noter que chaque élément d'un descripteur est un élément de base et qu'il peut lui-même devenir élément générateur.

Un descripteur peut donc devenir complexe, c'est-à-dire représenté par une arborescence à plusieurs niveaux.

- "la racine d'une description" est définie comme tout élément générateur qui est élément du descripteur :



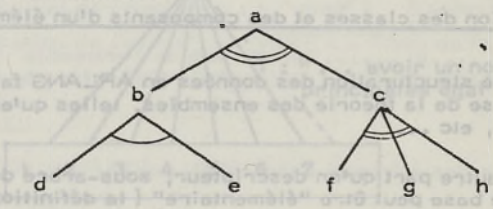
- le "vocabulaire" est constitué de tous les éléments de base, c'est-à-dire tous les mots utilisés par l'architecte.

322 - Représentation arborescente des descriptions

3221 - Elément de base ou élément générateur

Un élément de base ou élément générateur sera localisé au niveau des noeuds d'une arborescente de descriptions

Notation schématique



3222 - Descripteur

Un descripteur d'un élément x sera un ensemble d'éléments de base ou éléments générateurs ayant comme antécédent commun x (définition du sous-arbre).

Exemple :

{d, e} est un descripteur de b; d, e sont des éléments du descripteur de b

{b, c} est un descripteur de a; b, c sont des éléments du descripteur de a

{d, e, f, g, h, b, c,} est un descripteur de a; d, e, f, g, h, b, c sont des éléments du descripteur de a

3223 - Description

C'est un élément générateur associé à l'un de ses descripteurs.

Exemple :

b (d, e) est une description

a ((b (d, e) , c ((f, g, h)))) également

dans l'exemple, a est la racine de la description
a, b, c, d, e, f, g, h sont des éléments du vocabulaire V

3224 - Niveau et profondeur de description

- On appelle niveau de description d'un objet le niveau de l'objet dans la structure arborescente.

- On appelle profondeur de description d'un arbre, le niveau maximal de ses feuilles.

323 - Définition des classes et des composants d'un élément de base

La méthode de structuration des données en ARLANG fait appel à des notions de base de la théorie des ensembles, telles qu'ensemble, relation, classes, etc ...

Rappelons d'autre part qu'un descripteur, sous-arbre décrivant (ou non) un élément de base peut être "élémentaire" (la définition sera précisée dans ce paragraphe) ou "non élémentaire".

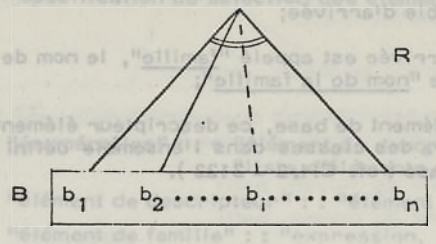
"descripteur" : : = "descripteur élémentaire" £ "descripteur non élémentaire" (1)

3231 - Classes Descripteur "de type ou"

(1) Nous débutons dans ce paragraphe l'exposé de la syntaxe du langage ARLANG. Cette syntaxe utilise le principe de la notation de BACCHUS exposé en annexe 12.

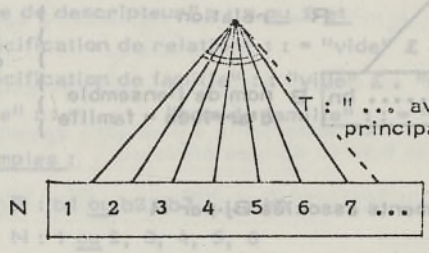
- ARLANG :

En ARLANG, les classes pourront être définies grâce à l'utilisation de descripteur élémentaire "de type ou", c'est-à-dire les descripteurs élémentaires munis de l'opérateur sémantique ou.



Exemple :

Un ensemble d'appartements pourra être partitionné en plusieurs classes de la façon suivante :

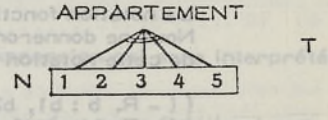


T est une relation de APPARTEMENT vers N

1, 2, 3, 4, 5 sont les images des éléments de APPARTEMENT par la relation T.

Les classes ainsi formées seront :

APPARTEMENT : 1
" : 2
" : 3
" : 4
" : 5



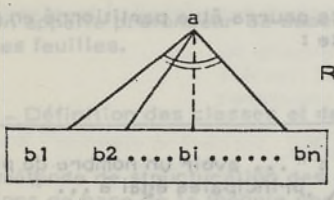
- Définition

Un descripteur élémentaire de type ou est défini par :

- une relation R d'un ensemble de départ non défini a priori vers un ensemble d'arrivée;
- l'ensemble d'arrivée est appelé "famille", le nom de cet ensemble étant le "nom de la famille";
- attaché à un élément de base, ce descripteur élémentaire de type ou définira des classes dans l'ensemble défini par l'élément de base (cf. CH.2 - 2122).

- Notation

Notation schématique :

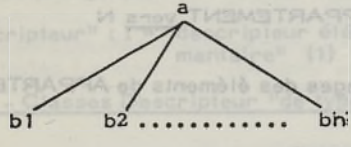


élément de base associé au descripteur

R relation) descripteur

B nom de l'ensemble d'arrivée = famille)

ensemble des éléments associés Bj par R



sous-ensembles-classes définies sur l'élément associé a

- Notation fonctionnelle :

La notation fonctionnelle sera définie ultérieurement (Nous ne donnerons dans ce paragraphe qu'un exemple d'utilisation de cette notation :

((- R. b : b1, b2, ... bn))
((- T. N : 1, 2, 3, 4, 5, 6))

Le double parenthésage encadrant le descripteur indique son type (ou).

Notation syntaxique des descripteurs élémentaires (de type ou ou de type et)

"descripteur élémentaire" : : = "spécification de relation" "spécification de famille" : "spécification de sélection des éléments du descripteur"

"spécification de sélection des éléments du descripteur" : : = "énumération" £ "spécification de fonction de description"

"énumération" : : = "élément de descripteur" "type de descripteur" "liste d'éléments de descripteur"

"élément de descripteur" : : "élément de famille" £ nil

"élément de famille" : : "expression, 1, J" £ "variable élément de descripteur"

"variable élément de descripteur" : : = "variable, J" avec J numérique, logique, caractère, accès

"type de descripteur" : : = ou £ et

"spécification de relation" : : = "vide" £ - "indicateur de relation"

"spécification de famille" : : "vide" £ . "nom de famille"

"vide" : : = "nom de famille" : : = "identificateur"

Exemples :

- R. B : b1 ou b2, b3, ... bn
- T. N : 1 ou 2, 3, 4, 5, 6

Sémantique :

- la création d'un descripteur élémentaire se fera au moment de son affectation à une variable descripteur ou de son attachement à un élément de base (cf. CH 2 - 2)
- les familles utilisées pour créer des descripteurs élémentaires n'auront pas à être créées au préalable; elles le seront implicitement au moment de la création du descripteur.

Exemple :

- T. N : 1 ou 2, 3, 4, 5, 6 est une expression qui sera interprétée de la manière suivante :

1) si la famille N n'existe pas déjà, créer cette famille et ses éléments 1, 2, 3, ... 6

si la famille N existe ne créer que les éléments manquants

2) créer le descripteur cité

Observations :

- la spécification de famille peut être omise au moment de la création d'un descripteur. Si tel est le cas, les éléments de descripteurs énumérés seront placés dans le vocabulaire, qui fait office de famille "fourre-tout"

  Ex : - TYPE : grand ou moyen, petit

- la spécification de relation peut également être omise; la relation TYPE est alors implicitement affectée à ce descripteur élémentaire

  Ex : . DIM : grand ou moyen, petit
  : grand ou moyen, petit

- les éléments d'une même famille doivent être tous de valeurs différentes; ces valeurs peuvent être de type numérique, chaîne, logique ou accès; mais une famille contient toujours des éléments de même type (cf. déclarations CH.2 - 211).

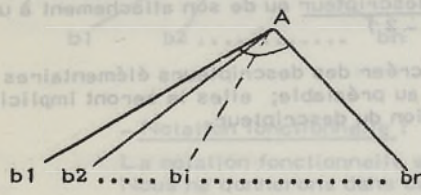
- les éléments d'un même descripteur élémentaire seront donc tous différents.

- un élément peut appartenir à plusieurs familles différentes.

- un même élément peut appartenir à plusieurs descripteurs élémentaires, de même type ou de type différent.

2332 - Eléments ou "composants" d'un élément de base - Descripteur de "type et" :

En ARLANG, les composants d'un élément de base pourront être définis grâce à l'utilisation de descripteur élémentaire "de type et", c'est-à-dire les descripteurs élémentaires munis de l'opérateur sémantique et

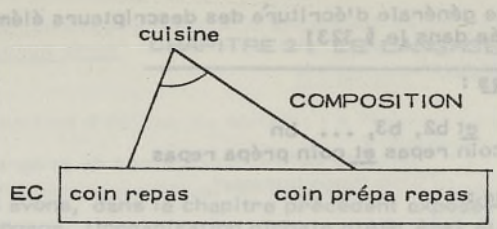


dans ce cas, la signification du descripteur muni de l'opérateur et est que chacun des éléments bi, i = 1, ... n de B appartiennent à l'élément de base A auquel le descripteur est associé

La relation ainsi spécifiée sera toujours appelée (différence fondamentale avec l'utilisation du descripteur de type ou) la relation de composition notée $\in$

$\forall$ bi$\in$B, bi$\in$ a
i = 1, ... n

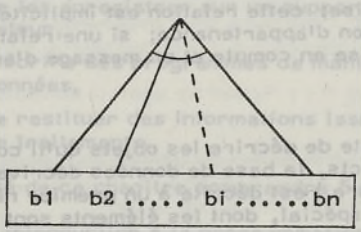
Exemple :



ces deux éléments sont ainsi des éléments de cuisine

- Notation

Notation schématique



la relation n'est pas spécifiée

B

élément de B appartenant à l'élément de base associé

Notation fonctionnelle

Exemple :

(B : b1, b2, ... bn)

(EC : coin repas, coin prépa repas)

Le simple parenthésage encadrant le descripteur indique son type (et)

Notation syntaxique :

La syntaxe générale d'écriture des descripteurs élémentaires a été exposée dans le § 3231

Exemples :

. B : b1 et b2, b3, ... bn
. EC : coin repas et coin prépa repas

Sémantique :

La sémantique est analogue à celle spécifiée dans le § 3231

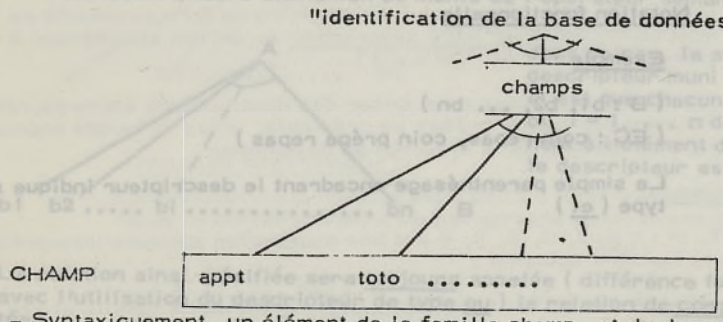
Observations :

Les mêmes observations qu'en § 3231 sont applicables. Seule la deuxième portant sur la spécification de relation est modifiée dans le sens suivant :

- la spécification de relation dans un descripteur élémentaire de type et doit être omise; cette relation est implicitement prise dans le sens d'une relation d'appartenance; si une relation est spécifiée, elle ne sera pas prise en compte et un message d'erreur sera édité.

3233 - Notion de "champ"

Afin de permettre à l'architecte de décrire les objets qu'il compte manipuler au sein de domaines distincts, la base de données décrivant un ensemble d'objets (et structurée en arbre) est décrite à un premier niveau (niveau 0) par un descripteur de type et spécial, dont les éléments sont éléments d'une famille particulière appelée champ.



- Syntaxiquement, un élément de la famille champ, et du descripteur ci-dessus, sera implicitement créé à chaque fois qu'un nouveau mot, qui n'est pas élément de cette famille et qui apparaît dans une racine de description, sera décrit (cf. CH.2 - 2121)

- L'accès aux descriptions de ces divers champs se fera en nommant le nom du champ, racine de description (cf. CH.2 - 2121).

CHAPITRE 2 : LE LANGAGE ARLANG

Nous avons, dans le chapitre précédent exposé, outre les objectifs généraux du langage, l'organisation globale d'ARLANG élaborée pour répondre à ces objectifs, ainsi que les éléments fondamentaux du métalangage ARLANG permettant une structuration cohérente de la description d'objets architecturaux ou urbanistiques.

Nous allons exposer dans cette partie le langage de programmation ARLANG; le langage permet :

- de constituer ou de modifier des bases de données structurées,
- de les enregistrer sur un support adéquat à leur traitement par ordinateur,
- d'écrire des programmes de manipulation et de traitement de ces données,
- de restituer des informations issues de la base de données ou résultats de traitements.

L'exposé de ce chapitre comprendra 5 points principaux :

1/ Une introduction à la programmation en ARLANG, exposant les divers principes adoptés quant à la conception de divers programmes et à leur enchaînement

2/ Un exposé sur les différents moyens d'enregistrer des données, soit dans une base de données, soit en unité centrale

3/ Un exposé sur les différents moyens de restituer des données

4/ Un exposé sur les diverses opérations de traitement de base que l'on peut effectuer sur des données ARLANG

5/ Un exposé sur les divers moyens de contrôler le déroulement d'un algorithme programmé.

Chacune des entités syntaxiques principales décrites seront analysées selon les points suivants :

- Rôle global
- Syntaxe et exemples syntaxiques
- Sémantique et exemples sémantiques
- Contraintes et options

1 - INTRODUCTION A LA PROGRAMMATION

Afin de donner le maximum de souplesse à la programmation, ARLANG a été muni de deux modes de programmation : le mode dit "mode bureau" et le mode dit "mode exécutable".

Syntaxe :

"programme arlang" : : = "suite de définition de blocs de programme" £ "bloc mode bureau"

11 - Programme en mode bureau :

- Le mode bureau permet d'écrire en mode entièrement conversationnel un programme afin d'assurer un échange immédiat d'informations, entre l'homme et le machine et vice-versa.

- D'autre part, c'est la seule façon de lancer l'exécution d'un programme en mode exécutable, appliqué à une base de données déterminée.

- Il permet de plus de modifier l'écriture d'un programme en mode exécutable (éditeur)

111 - Syntaxe

"bloc mode bureau" : : = "appel aux bases de données et de programmes" ; "chaîne d'instructions en mode bureau" "fin projet" £ "session édition de texte"

"instruction en mode bureau" : : = "instruction mixte" £ "instruction de modification de base de programme" £ "instruction de suppression automatique d'état"

"appel aux bases de données et de programmes" : : = "appel à la base de données" "appel à la base de programme"

1111 - Programmation conversationnelle

"appel à la base de données" : : = projet "nom de la base de données" : "liste de noms de base de données" £ projet "nom de la base de données" £ "vide"

"nom de la base de données" : : = "identificateur" "liste de clés"

"clé" : : = "identificateur" £ "nombre"

"instruction mixte" : : = "instruction de description" £ "instruction d'affectation" £ "instruction de mise en relation" £ "instruction de rupture de relation" £ "instruction de liaison" £ "instruction de rupture de lien" £ "instruction d'entrée-sortie" £ "appel de bloc"

"appel de bloc" : : = "appel de programme" £ "appel de fonction de description"

"instruction d'entrée-sortie" : : = "instruction d'entrée" £ "instruction de sortie"

1112 - Edition de texte :

"session édition de texte" : : = éditeur "identificateur de programme" ;
"chaîne d'instruction d'édition de texte"
fin éditeur

"instruction d'édition de texte" : : = "instruction de modification de texte"
£ "instruction de listage de texte"

"instruction de modification de texte" : : = "suppression d'une entité" £ "ajout d'une entité" £ "remplacement d'une entité par une autre"

"suppression d'une entité" : : = supprimer "numéro d'entité"

"ajout d'une entité" : : = ajouter "numéro d'entité" "entité"

"remplacement d'une entité par une autre" : : = remplacer "n° d'entité" par "entité"

"n° d'entité" : : = "nombre entier"

"entité" : : = "déclaration" £ "instruction arlang" £ "commentaire"

"instruction de listage de texte" : : = éditer "identificateur de programme" "spécification de fourchette d'entité"

"spécification de fourchette d'entité" : : = entre "n° d'entité" et "n° d'entité" £ "vide"

1113 - Base de programme :

"appel à la base de programme" : : = programme "nom de la base de programmes" : "liste de noms de base de programmes" £ programme "nom de la base de programmes" £ "vide"

"nom de la base de programmes" : : = "identificateur" "liste de clés"

"instruction de modification de base de programmes" : : = "modification" "tête de bloc"

"modification" : : = sauver £ oter

112 - Sémantique et observations :

Un bloc mode bureau a donc trois rôles :
- effectuer un certain travail sur une base de données
- éditer ou modifier le texte d'un bloc de programme (exécutable)
- modifier une base de programmes (sauvetage d'un programme, ôter un programme).

1121 - Travail sur une base de données : programmation conversationnelle

a - Définition de la base de données

Plusieurs cas peuvent se présenter :

- soit la base de données existe; on y fait appel pour y retirer, entrer ou traiter certaines informations.

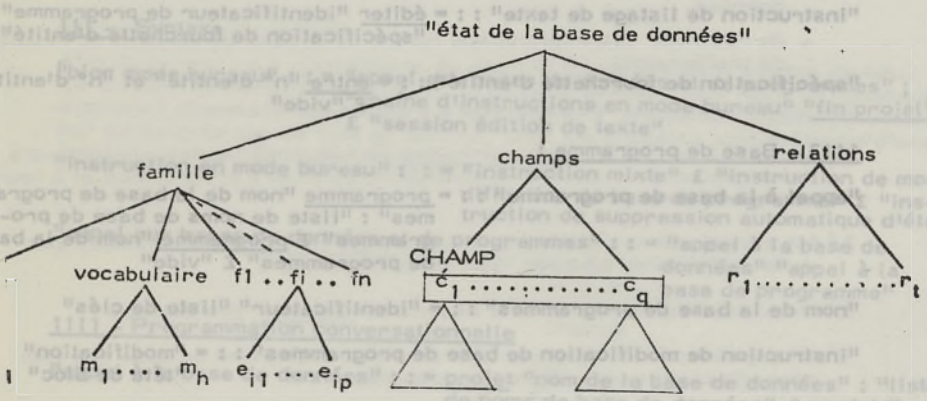
Le programmeur devra alors spécifier le nom de la base de données sur laquelle le programme en mode bureau qu'il va écrire s'exécutera

- soit la base de données doit être créée :

  - elle est créée par fusion d'autres bases de données : le programmeur donne donc un nom à la nouvelle base, et indique la liste des bases à fusionner. Les bases d'origine sont conservées dans leur état au moment de la fusion. La fusion agit par copie et fournit une base qui a la structure normalisée commune à toutes les bases de données ARLANG, et définie ci-dessous :

    - structure d'une base de données dans un état donné :

    sa structure sera identique à la structure des données ARLANG



La base contiendra de plus l'état des variables d'accès et des variables-élément de descripteur ainsi que l'état et la liste des fonctions de description attachées à certains de ses noeuds.

- le nom de la base de données incluera toutes les spécifications permettant de rendre l'accès à cette base public, privé ... Il jouera le rôle de clé d'accès

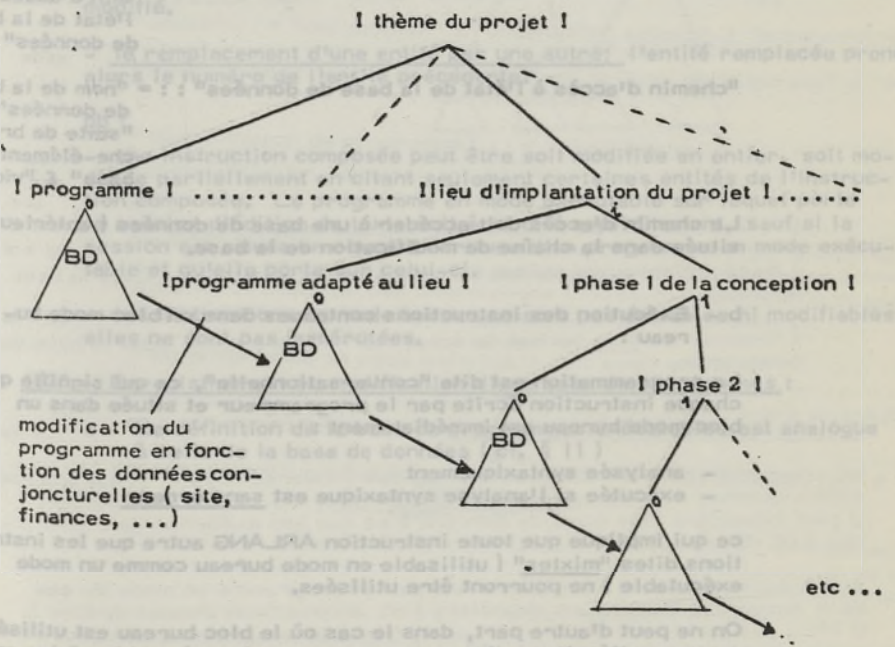
- la base de données est créée de toute pièce : le programmeur lui donne un nom

- notion d' "état" d'une base de données :

Le principe de sauvetage, entre plusieurs traitements sur une même base de données, des états intermédiaires de la base, est instrumenté en ARLANG de la manière suivante :

- le sauvetage est automatique; c'est seulement quand la place-mémoire fait défaut que le programmeur doit "purger" sa base

Sur l'exemple de l'évolution d'un projet, voyons comment la base peut évoluer :



De cette manière, on peut par exemple conserver tous les états de la base pendant le processus de conception, et ainsi revenir à certains états, faire d'autres modifications, etc ...

- l'accès à un état de la base se fait de la même manière que l'accès dans des champs ("chemin d'accès")

- toute modification de la base pourra être effectuée grâce aux instructions de modification de description ARLANG.

Il est fourni de plus une opération de "purge automatique" qui permet de ne conserver dans une base que le dernier état, ce dernier état prenant la place d'un état cité ou non. Si l'état n'est pas cité, il est pris implicitement comme le 1er état de toute la base.

Syntaxe :

"instruction de suppression automatique d'état" : : = substituer "chemin d'accès à l'état de la base de données"

"chemin d'accès à l'état de la base de données" : : = "nom de la base de données" "suite de branche-élément de base" £ "vide"

Le chemin d'accès doit accéder à une base de données (antérieure située dans la chaîne de modification de la base.

b - Exécution des instructions contenues dans un bloc mode bureau :

La programmation est dite "conversationnelle", ce qui signifie que chaque instruction écrite par le programmeur et située dans un bloc mode bureau est immédiatement :

- analysée syntaxiquement
- exécutée si l'analyse syntaxique est sans erreur

ce qui implique que toute instruction ARLANG autre que les instructions dites "mixtes" (utilisable en mode bureau comme un mode exécutable) ne pourront être utilisées.

On ne peut d'autre part, dans le cas où le bloc bureau est utilisé en cours d'écriture d'un programme en mode exécutable, faire appel (dans le bloc mode bureau) à ce programme exécutable en cours de codage, ainsi non plus qu'aux références (identificateur) qui y sont définies. D'autre part, toute séquence en mode bureau a un effet nul sur le codage du programme en cours d'écriture.

1122 - Session d'édition de texte :

Nous proposons un éditeur de texte simplifié dont les commandes font partie exclusivement d'un bloc en mode bureau.

Ces commandes permettent d'effectuer, sur un programme en mode exécutable :

- un listage des entités de ce programme, listage complet ou défini entre 2 bornes incluses.

Ce listage permet de plus d'expliciter le numérotage des entités affecté implicitement par l'interpréteur. Ce numérotage est utile pour les instructions d'éditeur de texte.

- une suppression d'entité, ou l'ajout d'une entité après une autre

La suppression et le rajout entraînent après l'exécution de l'instruction en mode bureau le renumérotage automatique des entités du programme modifié.

- le remplacement d'une entité par une autre; l'entité remplacée prend alors le numéro de l'entité précédente.

nb :

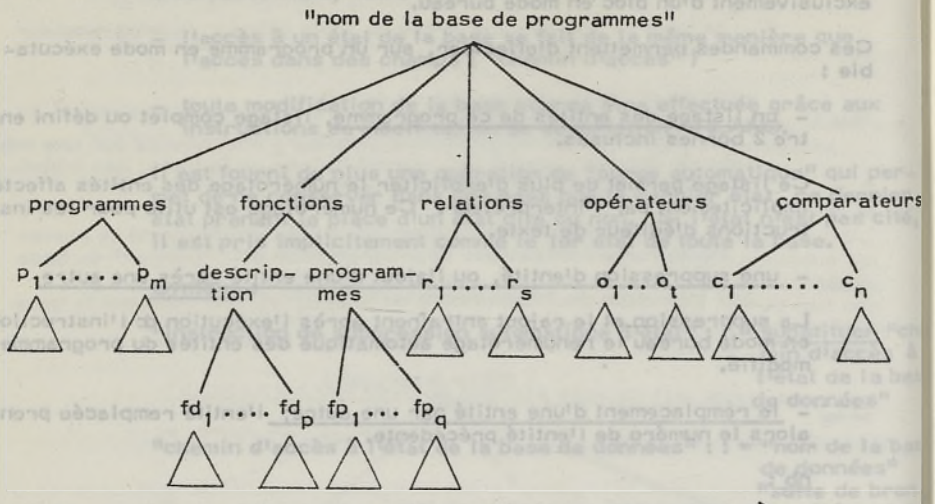
- une instruction composée peut être soit modifiée en entier, soit modifiée partiellement en citant seulement certaines entités de l'instruction composée. Le programme en mode exécutable sur lequel porte la session d'édition de texte doit être cité explicitement, sauf si la session est située en cours d'écriture d'un programme en mode exécutable et qu'elle porte sur celui-ci.

- les instructions en mode bureau ne sont pas éditables ni modifiables; elles ne sont pas numérotées.

1123 - Enregistrement ou modification d'une base de programmes :

a - La définition de la base de programmes exécutables est analogue à celle de la base de données (cf. § 11)

Chacune des bases de programmes créées ont la structure suivante :



b - Instructions de modification de base de programme

L'instruction sauve permet d'enregistrer un programme dans la base spécifiée.

L'instruction ôter permet de supprimer un programme dans la base de programme.

12 - Programme en mode exécutable :

Un programme en mode exécutable n'est pas conversationnel; il est analysé syntaxiquement et codé, enregistré en mémoire, mais n'est pas exécuté. Il peut cependant être écrit de manière à ce que son exécution soit interactive (cf. CH. 2 - 55).

Son exécution ne peut être lancée que par un programme en mode bureau dans lequel une instruction l'appelera (cf. instructions d'appel de bloc). Il n'y a pas de hiérarchie a priori entre blocs de programme. On peut donc rendre "maître" n'importe quel bloc.

Syntaxe :

"définition de bloc de programme" : : = "tête de bloc" "instruction composée"

"tête de bloc" : : = "identificateur de programme" "argument p" relation "tête de relation" £ "tête de fonction" £ opérateur "tête d'opérateur" £ comparer "tête de comparaison"

"argument p" : : = ("liste d'arguments") £ "vide"

"tête de fonction" : : = "tête de fonction de description" £ "tête de fonction programme"

"tête de fonction de description" : : = fonction "reste de fonction de description"

"tête de fonction programme" : : = fonction "identificateur" "argument p"

"instruction composée" : : = début "chaîne d'entités" fin

"entité" : : = "déclaration" £ "instruction arlang" £ "commentaire"

"instruction arlang" : : = "instruction" £ "bloc mode bureau" £ "instruction étiquettée"

"identificateur de programme" : : = "identificateur"

"argument" : : = "expression, 1, J"

nb :

On peut stopper le codage d'un bloc en passant à une série de programmes écrits en mode bureau; ce passage permet d'obtenir des informations ou de modifier des informations ("feed-forward") afin de guider la continuation de la programmation du problème en cours.

Tout programme en mode exécutable doit avoir un nom.

Nous verrons par la suite l'activation des différents types de programmes définis ci-dessus.

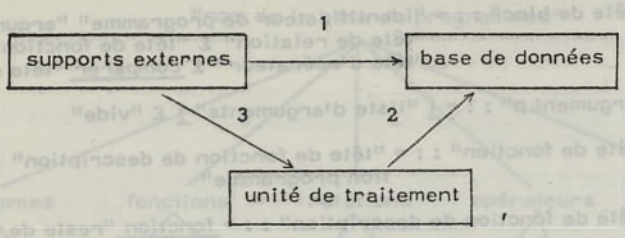
"instruction" : : = "instruction composée" £ "instruction mixte" £ "instruction algorithmique"

"instruction algorithmique" : : = "instruction itérative" £ "instruction conditionnelle" £ "instruction de branchement" £ "instruction d'interruption" et de reprise"

"instruction d'interruption et de reprise" : : = "instruction d'interruption" £ "instruction de reprise"

2 - DESCRIPTION DES INFORMATIONS

Ce paragraphe concerne toutes les entités du langage permettant d'effectuer des opérations dans le cadre du processus schématisé ci-dessous



- Les opérations (1) permettent d'enregistrer directement des données en provenance du monde extérieur à la machine dans une base de données

- Les opérations de type (2) permettent d'enregistrer dans une base de données les résultats de traitements de données exécutés préalablement.

- Les opérations de type (3) permettent l'enregistrement de données non incluses dans le contexte d'une base de données ARLANG, mais utilisées par un traitement sur cette base.

Les opérations (1) et (2) sont dites : "opérations d'enregistrement dans une base de données" d'informations en provenance du "supports externes" ou de l'unité de traitement.

Les opérations de type (3) seront appelées : "opérations d'enregistrement hors de la base de données".

21 - Les opérations d'enregistrement dans une base de données

Ce sont celles qui, situées dans un programme en mode bureau ou mode exécutable, permettent d'enregistrer directement des informations formalisées, dans le contexte de la base de données nommée en tête de programme.

Ces opérations seront décomposées en 2 types :

- les déclarations
- les instructions

211 - Déclarations :

Les déclarations ont pour rôles :

- de faire correspondre à un identificateur de variable ou de famille une structure donnée, permettant de contrôler la cohérence des traitements ultérieurement effectués sur ces structures.

- de définir la globalité ou la localité d'identificateurs de variable ou d'étiquette dans un programme

2111 - Syntaxe :

"déclaration" : : = "déclaration implicite" £ "déclaration explicite" £ "étiquette globale"

"déclaration implicite" : : = implicit "nature" "spécification de lettre"

"nature" : : = "nature variable" £ "nature famille"

"nature variable" : : = "localité" "structure et type de variable"

"localité" : : = "vide" £ local £ global

"structure et type de variable" : : = "structure"

"structure" : : = "structure élément de descripteur" £ "structure descripteur"
£ "structure accès"

"structure élément de descripteur" : : = "structure simple" £ "structure chaîne"

"structure simple" : : = "type"

"type" : : = entier £ réel £ logique

"structure chaîne" : : = chaîne

"structure descripteur" : : = descripteur

"structure accès" : : = accès

"nature famille" : : = famille "structure et type de famille"

"structure et type de famille" : : = "structure de famille"

"structure de famille" : : = "structure simple" £ "structure chaîne"

"spécification de lettre" : : = ("lettre" - "lettre") £ ("liste de lettres")

"lettre" : : = A £ B £ C £ .... £ Z £ a £ b £ ..... £ z

"déclaration explicite" : : = "nature" "liste d'identificateurs de variable ou de famille"

"identificateur de variable ou de famille" : : = "identificateur de variable" £ "identificateur de famille"

"identificateur de variable" : : = "identificateur"

"identificateur de famille" : : = "identificateur"

"identificateur" : : = "lettre" £ "identificateur" " lettre" £ "identificateur"
"chiffre" £ "identificateur"

"chiffre" : : = 0 £ 1 £ 2 £ 9

Exemple :

implicit local descripteur (D); (toute variable commençant par D est une variable descripteur locale)

implicit famille chaîne (C); (toute famille commençant par C est une famille dont les éléments sont des chaînes)

famille chaîne FC, TØTØ FC, TØTØ ont leurs éléments de type chaîne

global entier $D; la variable $D est une variable entière

2112 - Sémantique et observations :

a - Globalité et localité des identificateurs :

- les variables déclarées locales ont une portée "statique" réduite au bloc de programme dans lequel elles sont déclarées.

  Si la "localité" n'est pas précisée, les variables déclarées seront implicitement considérées comme globales

- Tout identificateur d'une base de données a une portée globale vis-à-vis de programmes qui se servent de cette base; ils sont par contre locaux à la base où ils apparaissent : un même identificateur apparaissant dans deux bases différentes peut donc avoir 2 significations différentes. Par contre, un même identificateur apparaissant dans deux blocs de programme différents travaillant sur la même base de données, représente la même chose.

b - Les déclarations peuvent se placer à n'importe quel point du programme, mais toujours avant la première occurence de l'identificateur dans le programme.

c - Structures

Une structure simple est réduite au type de la valeur : entier numérique, réel numérique, valeur logique (vrai ou faux)

Une structure chaîne définit des variables ou familles contenant des "chaînes"; une chaîne est une concaténation de caractères alpha numériques ou spéciaux.

Une structure accès définit des variables ou familles contenant des "accès" (cf. CH.2 - 2121) - (les accès sont toujours considérés comme multiple; si la syntaxe oblige le programmeur à utiliser un accès unique, il suffit qu'à l'exécution la variable en question ne contienne qu'un accès).

Une structure descripteur définit des variables contenant des descripteurs élémentaires ou non. Une famille ne peut pas contenir comme valeurs, des descripteurs (en particulier toute famille est donc homogène par rapport au type et à la structure de ses éléments). Seule la famille "fourre-tout", constituée par le vocabulaire, est hétérogène : aucune vérification de compatibilité des types ou structures ne sera donc effectuée au moment du traitement

sur ces éléments.

Spécification de lettre

implicit entier (X - Z) signifie que toutes les variables commençant par les lettres X, Y, Z, contiennent des valeurs de type entier

Longueur des identificateurs

- La longueur des identificateurs est limitée à 20 caractères
- Deux identificateurs consécutifs doivent être séparés par au moins un blanc
- Les identificateurs peuvent contenir des blancs, qui sont alors soulignés.

212 - Instructions

Les instructions permettant d'effectuer des opérations d'enregistrement dans une base de données sont toutes de type mixte. Avant d'exposer ces différentes instructions, nous devons exposer le moyen d'accéder volontairement à des noeuds d'une base de données.

2121 - La notion de "chemin d'accès"

Une fois indiquée la base de données sur laquelle on travaille, toute opération d'enregistrement, de modification ou de restitution d'indormations d'une base de données se fait dans le cadre des différents champs créés en cours de processus.

Il convient de localiser l'information à manipuler dans un champ en précisant son "contexte", c'est-à-dire le chemin partant de la racine de description d'un champ et aboutissant au noeud support de l'information désirée : ce chemin est appelé "chemin d'accès"

Un chemin d'accès peut être unique ou multiple selon que l'on veut accéder à un seul noeud (un seul contexte) ou à plusieurs noeuds présentant des caractéristiques communes (même valeur, même famille, ...)

"chemin d'accès" : : = "chemin d'accès unique" £ "chemin d'accès multiple"

Dans tous les cas cependant, on ne pourra accéder qu'à des éléments de base situés dans le même champ.

a - Chemin d'accès unique :

Un chemin d'accès unique permet d'accéder à un élément de base dans un champ donné

Syntaxe :

"chemin d'accès unique" : : = "racine de description" £ "racine de description" "suite de branche-élément de base"

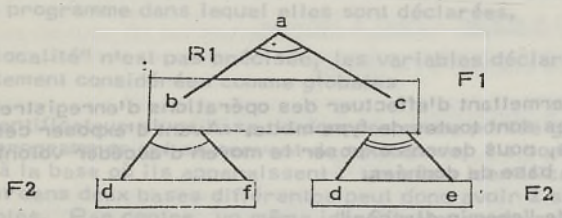
"racine de description" : : = champ : "élément de la famille champ"

"élément de la famille champ" : : = "élément de famille"

"branche-élément de base" : : = "spécification de relation" "spécification de famille" : "spécification unique d'élément de base"

"spécification unique d'élément de base": : = "élément de famille"

Soit le champ décrit en ARLANG de la façon suivante :



Un chemin d'accès unique à l'un des éléments d sera par exemple :

CHAMP : a - R1. F1; b. F2 : d

Sémantique :

L'évaluation d'un chemin d'accès donne comme résultat l'adresse du noeud cité en fin de chemin (d dans l'exemple).

Observations :

- A condition qu'il n'y ait pas d'ambiguité, c'est-à-dire si l'on n'a pas attaché au même élément générateur, un autre élément de même nom appartenant à un autre descripteur élémentaire (sinon l'accès est ambigu, une erreur est alors éditée), on peut omettre la spécification de famille ou/et la spécification de relation.

- Noter qu'au niveau de l'expression d'accès, on ne fait pas de différence entre un noeud-élément d'un descripteur élémentaire de type ou et un noeud-élément de descripteur élémentaire de type et.

Une autre forme syntaxique sera plus tard proposée rendant compte de ce caractère.

b - Chemin d'accès multiple :

Le rôle d'un chemin d'accès multiple est d'accéder simultanément à plusieurs noeuds dans un champ donné, ces noeuds ayant certaines caractéristiques communes

Ces caractéristiques communes peuvent :

- avoir le même nom
- appartenir à un même ensemble : famille, descripteur élémentaire ...
- être les parties d'un même ensemble.

Syntaxe :

"chemin d'accès multiple" : : = "primaire J5" / "fin de chemin d'accès multiple" "spécification de niveau"

"fin de chemin d'accès multiple" : : = "suite de branche-élément de base" £ "suite de branche-élément de base facultatif" "extrémité" £ "spécification d'accès aux classes"

"primaire J5" : : = "variable J5" £ "indicateur de fonction J5" £ ("expression, 1, J5") £ nil £ "chemin d'accès"

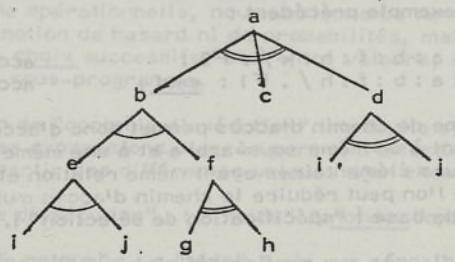
"branche-élément de base facultatif" : : = "vide" £ "branche-élément de base"

Sémantique :

- Chemin d'accès à des noeuds d'un sous-arbre portant le même nom

Syntaxiquement, la fin du chemin d'accès se présentera sous la forme d'une "suite de branche-élément de base", le "primaire J5" pouvant représenter un chemin unique ou multiple.

Exemple :



(1) champ : a : b : f / - R1. F1 : i

(2) champ : a : b : f / - R1 : i

(3) champ : a : b : f / : k / : i

(4) champ : a : b : f / : i accès à tous les noeuds de valeur i dans le sous-arbre décrivant a : b : f

On voit donc qu'en jouant sur la spécification ou non de la relation et de la famille, et sur la qualité (unique ou multiple) de la primaire J5, on peut spécifier des accès différents.

En jouant sur l'existence ou non dans le chemin de la spécification de relation et de famille, on peut donc accéder à des éléments de même nom et :

- appartenant ou non à la même famille
- appartenant ou non à des descripteurs élémentaires de même relation
- appartenant ou non à des descripteurs élémentaires de même relation et de même famille

- Chemin d'accès à des noeuds appartenant à un même ensemble

La fin du chemin d'accès se présentera sous la forme d'une :

"suite de branche-élément de base facultatif" "extrémité"

"extrémité " : : = "spécification de relation" "spécification de famille" : "étoile"

"étoile" : : = * "spécification de sélection"

"spécification de sélection" : : = "vide" £ sauf "liste d'éléments de base"

"élément de base" : : = "élément de descripteur"

Exemple :

Dans l'exemple précédent :

champ : a : b : f : h : k / . F2 : accès à i

champ : a : b : f : h / . F1 : sauf j accès à i et k

Cette forme de chemin d'accès permet donc d'accéder à des éléments appartenant à un même sous-arbre et à une même famille, ou à des descripteurs élémentaires ayant même relation et même famille. Noter que l'on peut réduire le chemin d'accès multiple qu'à certains éléments de base ("spécification de sélection").

- Chemin d'accès aux parties définies sur un ensemble :

La fin du chemin d'accès se présentera sous la forme d'une "spécification d'accès aux partitions"

"spécification d'accès aux partitions" : : = "étoile"

Sur l'exemple du (1), le chemin champ : a / permettra d'accéder aux noeuds b, c, i, j (i, j descendants de d).

nb :

Le programmeur peut avoir besoin de spécifier, dans ces différentes façons d'accéder à des noeuds de l'arbre d'un champ, des niveaux de recherche dans l'arbre; il spécifiera alors, dans l'écriture du chemin d'accès, un paramètre de niveau :

"spécification de niveau" : : = "vide" £ entre "expression 1, entière", "expression 1, entière" £ jusqu'à "expression 1, entière" £ à partir de "expression 1, entière"

- Programmation non déterministe et notion de backtracking

On oppose en recherche opérationnelle l' "aléatoire" au "déterministe" : "un univers (une situation, un problème) est dit déterministe si le hasard n'intervient pas dans le cours du processus"... "Au contraire, dans un univers aléatoire, l'évolution des phénomènes est soumise à des lois de probabilités" (définition tirée du "Dictionnaire de l'Informatique", André LE GARFF).

En programmation (informatique), le "non-déterminisme" consiste, à partir d'un certain point d'exécution du programme, à générer plusieurs branches de programmes qui s'exécutent indépendamment et successivement.

L'opération de "backtracking" consiste, au moment du retour au point de décomposition du programme en plusieurs branches, et avant d'effectuer l'exécution d'une autre branche, à replacer l'état du programme (variables, données ...) dans l'état initial (au point de décomposition).

Donc, contrairement au sens donné à l'adjectif "déterministe" en recherche opérationnelle, nous utilisons ce terme sans faire intervenir la notion de hasard ni de probabilités, mais seulement la notion de choix successifs, soit entre valeurs, soit entre différents corps de sous-programme.

La notion de "contrôle d'exécution" introduite syntaxiquement au niveau des expressions d'accès permet d'effectuer du non-déterminisme à partir des différentes valeurs d'un chemin d'accès multiple.

"contrôle d'exécution" : : = "vide" £ nondet "option de retour"

"option de retour" : : = "vide" £ sb

- si le "contrôle d'exécution" est "vide", l'exécution du programme est déterministe par défaut

- si l'option de retour est spécifiée sb (sans backtracking), la

base de données conserve à chaque retour les modifications effectuées lors de l'exécution d'une branche de programme, mais les variables sont remises dans l'état initial.

Nous n'approfondirons pas cette notion de programmation non-déterministe en ARLANG. Elle fera l'objet des extensions du langage.

2122 - Les instructions de "description"

'Décrire" un élément de base signifie lui attacher un descripteur élémentaire ou non élémentaire. On généralisera ce terme à toute opération modifiant le descripteur d'un élément de base ou élément générateur.

Un élément de famille voit son sens précisé quand il devient élément de base (il est alors précisé par le "contexte" dans lequel il se situe), puis élément générateur (il est alors précisé par le descripteur qui le décrit). C'est la conjonction de ces 3 informations : contexte - élément générateur - descripteur qui constitue ce que l'on appelle un "OBJET-ARLANG".

Syntaxe :

"instruction de description" : : = "instruction d'attachement" £ "instruction de détachement" £ "instruction de suppression d'éléments de base" £ "instruction de rajout d'éléments de base"

"expression de descripteur" : : = "expression, 1, descripteur"

a - Attachement d'un descripteur à un noeud :

L'instruction décrite ici permet de décrire un élément de base ou un élément générateur défini par une "expression d'accès"

Syntaxe :

"instruction d'attachement" : : = attach "contrôle d'exécution" "expression 1, accès"; "liste d'expression du descripteur"

Exemple et sémantique :

Soit $CA une "variable d'accès"

$DA 11 une variable contenant un descripteur "élémentaire" de type ou

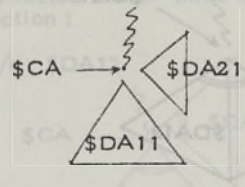
$DA 21 une variable contenant un descripteur élémentaire de type é

Ecrire :

attach $CA; $DA 11 aura pour conséquence la description par le descripteur contenu dans $DA1 du ou des noeu auxquels on accède par le chemin d'accès (unique ou multiple) contenu dans $CA.

attach $CA; $DA 21 aura une signification analogue

Si $CA est unique, l'exécution de ces 2 instructions aura par exemple pour résultat :

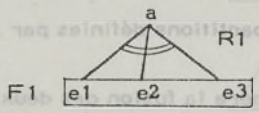


Ceci dans le cas simple où le noeud auquel on accède n'a pas été déjà décrit.

- Signification de l'attachement d'un descripteur à un noeud, élément d'un descripteur élémentaire de type ou de type et

  • descripteur de type ou :

  Nous avons dit qu'un descripteur de type ou permettait de décrire les partitions d'un ensemble-élément de base :

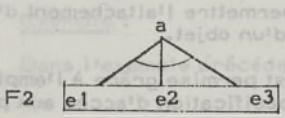


.... a-R1.F1 : e1
.... ------- : e2 sont les 3 parties de A
.... ------- : e3

  Attacher un descripteur de type ou ou et à l'une de ces parties, c'est donc décrire celle-ci, c'est-à-dire l'objet a qui a une certaine qualité définie ppar e1

  • descripteur de type et :

  Un descripteur de type et attaché à un élément de base définit les éléments de l'ensemble.



... F2 : e1
... F2 : e2 sont les éléments de a
... F2 : e3

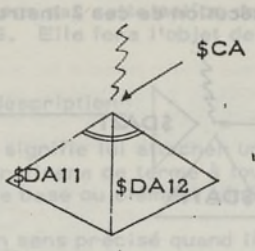
  Attacher un descripteur de type et à l'un de ces éléments, c'est décrire l'élément cité; il y a donc rupture de sens (on ne parle plus du même élément).

- notion de "fusion"

On appelle "fusion" de deux ou plusieurs descripteurs élémentaires de même type l'attachement de ces descripteurs à un même noeud.

Exemple :

- attach $CA; $DA11; $DA12 :



. $DA11 et $DA12 "fusionnent", c'est-à-dire que le noeud auquel ils sont attachés est partitionné par les éléments du descripteur contenu dans $DA1 et par ceux du descripteur contenu dans $DA2

. on pourra alors soit manipuler les partitions définies par $DA1, soit celles définies par $DA2

- attach $CA; $DA21; $DA22; entraînera la fusion des deux descripteurs élémentaires de type et contenus dans les 2 variables $DA21 et $DA22, l'ensemble des éléments définis par les deux descripteurs devenant alors éléments de l'ensemble porté sur le noeud.

Il convient de noter que l'on peut ainsi rajouter à volonté, successivement, un ensemble de partitions ou un ensemble d'éléments en rajoutant un ou plusieurs descripteurs élémentaires.

- Notion de "mise en facteur"

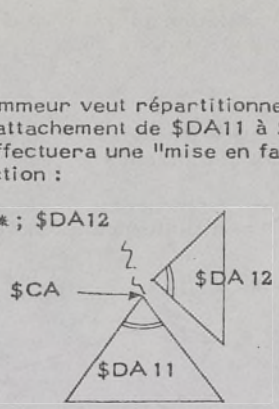
Le rôle d'une "mise en facteur" est de permettre l'attachement d'un descripteur à l'ensemble des partitions d'un objet.

Syntaxiquement, cette mise en facteur est permise grâce à l'emploi d'un chemin d'accès multiple contenant une spécification d'accès aux partitions (cf. CH 2 - 2121.b).

Exemple :

Si le programmeur veut répartitionner chacune des partitions définies par l'attachement de $DA11 à $CA, grâce à un descripteur $DA12, il effectuera une "mise en facteur" de $DA12 sur $DA11 par l'instruction :

attach $CA/* ; $DA12



dont le sens est :

Il convient de noter que tout descripteur élémentaire de type et est toujours en facteur par rapport aux partitions dont il définit les éléments. La mise en facteur sera automatique.

La notion de "sens" d'une arborescence dans laquelle des descripteurs ont été mis en facteur est importante : toute manipulation d'arborescence se fera sur l'arborescence dite "interprétée", c'est-à-dire où l'on ne trouvera plus de mise en facteur.

Exemple :

Dans l'exemple précédent

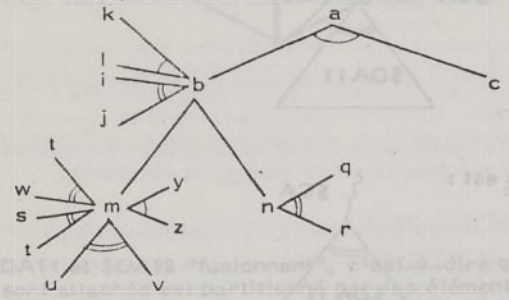
. l'opération d'interprétation est faite selon des règles strictes, ôtant ainsi toute ambiguité dans le sens d'une mise en facteur :

les 2 règles sont :

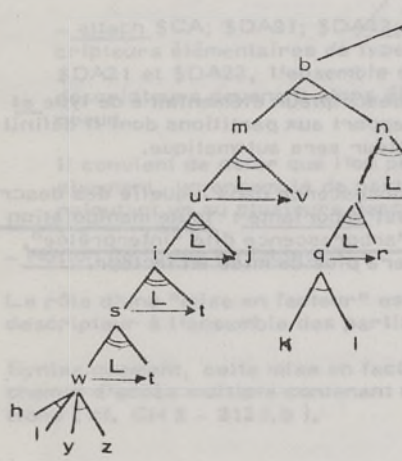
- l'ordre d'interprétation est indépendant de l'ordre selon lequel les descripteurs ont été attachés

- pour un noeud donné, l'interprétation des descripteurs de type ou se fait antérieurement à l'interprétation des descripteurs de type et

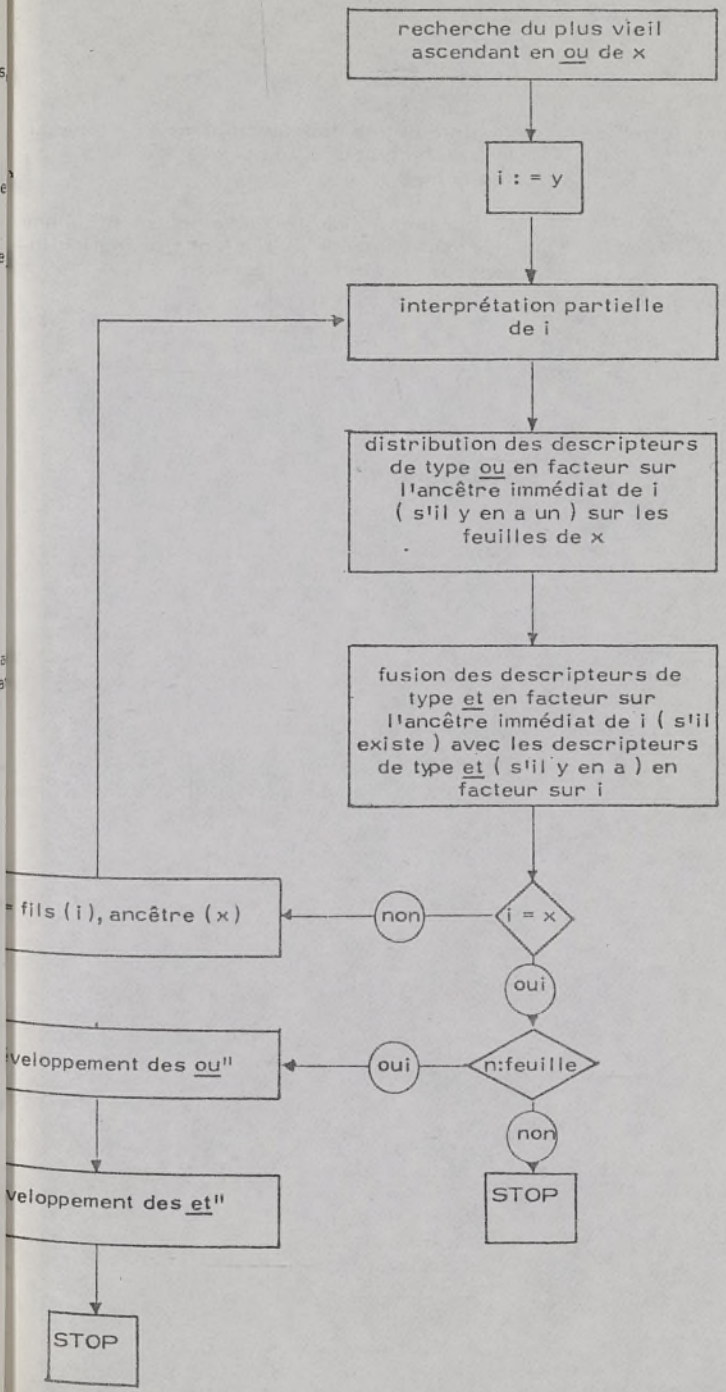
Soit l'exemple suivant :



L'interprétation totale de cet arbre (interprétation totale de chaîne de ses feuilles), grâce à l'algorithme présenté en page suivante, est l'arborescence suivante :



Algorithme d'interprétation totale d'un noeud (et d'une feuille en particulier)x:

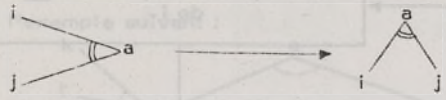


On appelle :

"interprétation partielle" de i : la distribution des descripteurs de type ou mis en facteur sur le noeud i, les uns sur les autres

"développement" : par analogie avec le "développement" d'une expression algébrique, c'est l'opération inverse de la "mise en facteur"

Exemple :



développement

nb :
Une mise en facteur de descripteur ou peut être spécifiée, même si l'élément de base considéré n'a pas encore été décrit : on anticipe en quelque sorte sur les descriptions futures (il en est de même pour tous les descripteurs de type et)

Cela vient du fait que le sens d'une description avec mise en facteur est toujours potentielle; il n'est résolu qu'au moment du besoin puis redevient potentiel. Ceci aura certaines conséquences au niveau des accès et des modifications de description.

b - Détachement d'un descripteur d'un noeud :

Syntaxe :

"instruction de détachement" : : = détacher "chaîne de descripteur et accès"

"descripteur et accès" : : = "contrôle d'exécution" "expression 1, accès"; "liste d'expression 1, descripteur" £ "fonction consultative de descripteur d'un accès"

Sémantique :

. Si l'on supprime un descripteur attaché à un noeud, on supprime du même coup :

- l'ensemble des descripteurs attachés à chacune des feuilles du descripteur spécifié
- si les mises en facteur, portant sur le descripteur détaché ou ceux implicitement détachés, ont été effectuées, ces mises en facteur sont conservées (en vertu de la "potentialité" définie ci-dessus), à condition bien sûr que leur "point d'attache" soit de niveau supérieur ou égal au niveau de l'élément de base duquel on détache le descripteur.
- toutes relations (cf. CH. 2 - 2123) mises en place et faisant référence à au moins un noeud du descripteur détaché sont perdues
- tout accès à des éléments du descripteur détacha devient impossible; les variables d'accès contenant ces chemins les contiendront toujours mais n'accéderont à rien.

c - Instruction de suppression d'élément de base :

"instruction de suppression d'élément de base" : : = supprimer "contrôle d'exécution" "liste d'expression 1, accès"

Sémantique :

- la suppression portera sur l'ensemble des noeuds accédés
- la sémantique définie en b est également valable.

d - Rajout d'éléments de base

"instruction de rajout d'élément de base" : : = rajouter "chaîne de lots

"lot" : : = "descripteur élémentaire et accès" ; "liste d'éléments de famille"

"descripteur élémentaire et accès" : : = "contrôle d'exécution" "expression 1, accès" ; "liste d'expression 1, descripteur élémentaire" £ "fonction consultative de descripteur élémentaire d'accès"

"expression, 1, descripteur élémentaire" : : "expression, 1, descripteur"

Sémantique :

- Si des mises en facteur ont été effectuées en amont des éléments rajoutés, les éléments rajoutés porteront les mises en facteur

- Si des variables d'accès permettent d'atteindre des éléments portant des noms qui ont été rajoutés, on ne pourra pas accéder aux éléments rajoutés. Il conviendra de réaffecter la variable.

2123 - Relations :

Nous avons vu dans le métalangage comment, grâce à la structure arborescente munie des opérateurs ou et et, nous pouvions spécifier 2 types de relation

1) les relations d'appartenance d'un ou plusieurs éléments à un ensemble (type et)

2) les relations d'un ensemble vers un autre (type ou) créant un partitionnement en classes de l'élément de base décrit.

Nous avons introduit une autre instruction, permettant de mettre en relation deux noeuds, au moment de la description d'objets architecturaux afin de pouvoir spécifier des contraintes communes à divers objets ou éléments de ces objets.

En ARLANG, en complément de l'instruction permettant de lier deux noeuds par une relation donnée, on pourra :

- définir le "sens" (la sémantique à d'une relation) (cf. b)
- tester l'existence de noeuds liés par une relation (cf. CH.2 - 3113 à
- tester la validité des descriptions de 2 noeuds liés par une relation (cf. CH.2 - 3113)

Nous n'examinerons dans ce chapitre que l'instruction de mise en relation et de définition de son "sens".

a - Relations quelconques

"instruction de mise en relation" : : = "indicateur de relation"

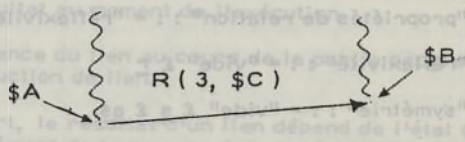
"indicateur de relation" : : = "tête de relation"

Exemple :

R $A, $B (3, $C) est une instruction mettant en relation les noeuds contenus dans les 2 variables d'accès $A et $B, la relation ayant pour paramètres effectifs 3 et $C.

Sémantique :

Les arguments de la relation sont des paramètres effectifs, en même nombre que ceux de la définition, de même type et de même ordre



Une relation est toujours binaire et orientée; les propriétés de certaines relations (réflexivité, symétrie, transitivité) induisent certains types de relation (ordre, préordre, équivalence) pourront et devront être spécifiées au moment de la définition de la relation.

nb :
Afin de pouvoir supprimer des relations liant deux noeuds en cours de programme, nous avons introduit une instruction de "rupture de relation"

"instruction de rupture de relation" : : = rompre "indicateur de relation
£ rompre relation "paire d'arguments d'accès"

Exemple :

rompre relation $A, $B aura pour effet de supprimer toutes les relations quelles qu'elles soient entre les noeuds définis par $A et ceux définis par B.

b - Sémantique d'une relation

Définir le "sens" d'une relation en ARLANG, c'est écrire un bloc de programme constitué d'une chaîne d'instructions dont l'activation donne comme résultat minimal une valeur logique vrai ou faux.

Syntaxe :

"définition de bloc de programme" : : = "tête de bloc" "instruction compilée"

"tête de bloc" : : = ... £ relation "tête de relation" £ ...

"tête de relation" : : = "nom de relation" "arguments de relation"

"nom de relation" : : = "identificateur"

"arguments de relation" : : = "propriétés de relation" "paire d'arguments d'accès" " arguments p"

"argument p" : : = "vide" £ ("liste d'arguments")

"propriétés de relation" : : = "réflexivité" £ "symétrie" £ "transitivité"

"réflexivité" : : = "vide" £ r

"symétrie" : : = "vide" £ s £ as

"transitivité" : : = "vide" £ t

"paire d'arguments d'accès" : : = "contrôle d'exécution" "expression accès" "contrôle d'exécution", "expression1, accès"

Sémantique :

- Une relation est toujours définie entre deux noeuds cités par la paire d'arguments d'accès

- Des spécifications supplémentaires peuvent être indiquées dans les arguments

- La spécification des propriétés d'une relation permet de contrôler la pertinence de la mise en relation de deux noeuds vis-à-vis des mises en relation précédentes sur la base de données et de générer automatiquement certaines mises en relation à partir d'autres.

c - Une relation particulière : le "lien"

Le lien est une notion ARLANG qui a été introduite dans le but de permettre au programmeur de rendre compte d'une identité (partielle ou totale) de descripteur entre deux éléments de base ou éléments générateurs. En cela, c'est une relation particulière fournie au programmeur, dont le "sens" est prédéfini.

- Syntaxe et sémantique du lien

"instruction de liaison" : : = lier "expression 1, accès unique", "contrôle d'exécution" "expression 1, accès"

"expression d'accès unique" : : = "chemin d'accès unique" £ "variable d'accès"

La variable d'accès ne doit contenir qu'un seul accès - le contrôle est effectué par l'interpréteur.

Le résultat de l'exécution d'une instruction de lien peut être décrit en 2 phases :

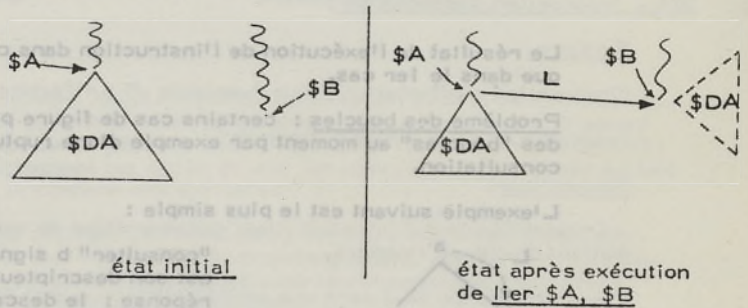
1°) le résultat au moment de l'exécution

2°) l'influence du lien au cours de la partie programme qui suit l'instruction de lien.

D'autre part, le résultat d'un lien dépend de l'état dans lequel se trouve l'élément de base auquel on accède par la 2me expression d'accès.

De façon générale "lier $A, $B" signifie décrire $B comme $A

1er cas : le cas de figure est le suivant :



état initial

état après exécution de lier $A, $B

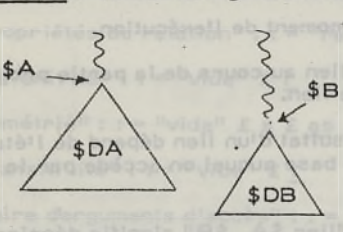
1°) L'exécution de l'instruction lier $A, $B entraîne la mise en facteur du descripteur $DA de $A sur $B

nb : "B n'est pas décrit" s'entend interprétation des mises en facteur comprise. Donc, quand on a exécuté l'instruction lier $A, $B, on peut accéder à des éléments du descripteur $A ($DA) aussi bien par une expression d'accès passant par le noeud $B que par le noeud $A. Par contre $DA ne peut être modifié dans le contexte de $B; dans le contexte de B, on ne peut donc que consulter $DA.

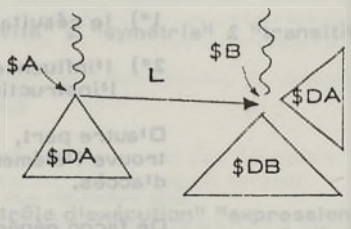
2°) L'instruction de lien est d'autre part potentielle, c'est-à-dire que tant que $A et $B ne sont pas déliés (cf. 2), toute modification du descripteur de $A ($DA) et effectuée dans le contexte de $A entraîne une modification de ce descripteur dans le contexte de $B (mais le contraire n'est pas vrai, c'est-à-dire que si l'on modifie $DA dans le contexte de $B, $DA dans le contexte de $A n'est pas modifié).

D'autre part, $B peut être décrit, sans affecter $DA, de même façon que si $DA était un descripteur mis en facteur sur $B indépendamment du lien.

2me cas : le cas de figure est le suivant :



état initial

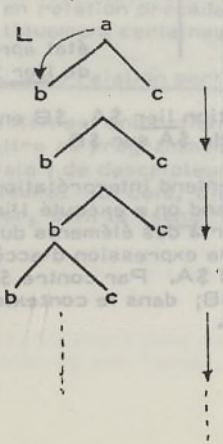


état après exécution de l'instruction lier A, B

Le résultat de l'exécution de l'instruction dans ce cas est le même que dans le 1er cas.

Problème des boucles : certains cas de figure peuvent engendrer des "boucles" au moment par exemple d'une rupture de lien ou d'une consultation.

L'exemple suivant est le plus simple :



"consulter" b signifie demander quel est son descripteur par exemple : réponse : le descripteur de b ($DB) c'est, b étant lié à a

mais dans ce descripteur, b est décrit, puisque lié à a. Donc :

mais dans ce nouveau descripteur, b est décrit, puisque lié à a ..... !

et ainsi de suite jusqu'à l'infini

- Rupture de lien

L'instruction de rupture de lien permet de rendre indépendants les descripteurs de 2 éléments, précédemment liés

Syntaxe :

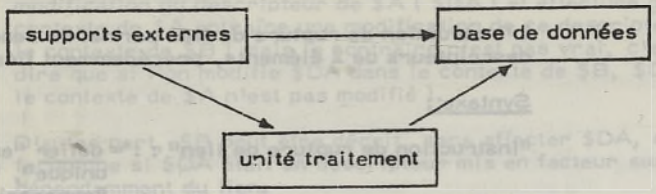
"instruction de rupture de lien" : : = délier "expression 1, accès unique", "contrôle d'exécutio "expression d'accès"

Sémantique :

- L'exécution de l'instruction de rupture de lien (par exemple délier $A, $B) va entraîner la recopie en $B du descripteur de $A, interprétation des mises en facteur effectuées en $A, dans l'état où il se trouve, au moment de la rupture du lien.

- Les deux descripteurs sont alors rendus indépendants et la modification de $DA dans $A n'entraînera plus la modification de $DA dans $B.

22 - Opérations d'affectation



Dans le schéma ci-dessus, les opérations d'affectation permettant d'enregistrer des données et de les adresser par un identificateur, s'appliquent aux 3 opérations indiquées :

- enregistrement de données externes ou de l'unité de traitement dans la base de données (variables éléments de descripteur)

- enregistrement de données dans l'unité de traitement

Les opérations d'affectation (d'une valeur à une variable) seront permises en ARLANG grâce à l'emploi de deux types d'instructions : les instructions d'affectation et l'instruction d'entrée.

221 - Instruction d'affectation :

Syntaxe :

"instruction d'affectation" : : = "variable J" : = "expression 1, J"

"variable J" : : = $ "identificateur" J∈{numérique, chaîne, logique, accès}

Sémantique :

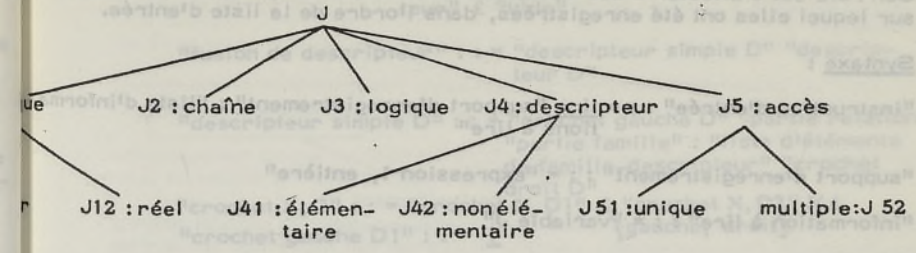
- L'identificateur J introduit n'est qu'un artifice de syntaxe nous permettant d'alléger l'exposé.

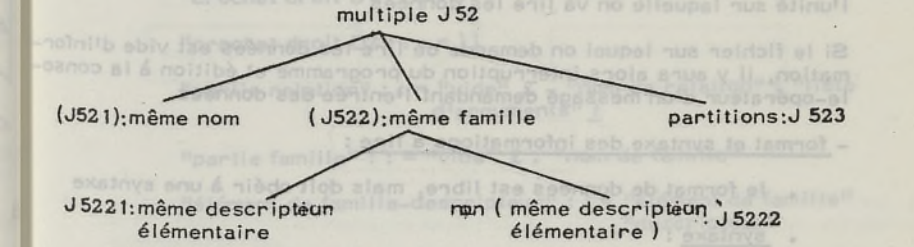
Toute occurence de J dans une entité syntaxique doit être remplacée par l'une de ses valeurs définie par l'arborescence ci-dessous et de façon homogène dans toute l'entité.

Exemple :

"variable J11" : : = "expression 1, J11 "

J représente la structure et le type de la variable ou de la valeur considérée (cf. CH.2 - 211)





- Une variable aura toujours comme contenu le résultat de l'évaluation de l'expression au moment de l'affectation. Ceci est particulièrement important pour les variables d'accès (J5) et variable-descripteur (J4) : toute modification sur les accès ou sur les descripteurs n'auront aucune influence sur le contenu des variables.

- Une variable d'accès (J5) contiendra toujours l'adresse du ou des noeuds calculée par évaluation du chemin d'accès. De ce fait, la comparaison de deux variables d'accès sera la comparaison d'adresses aux noeuds et non l'égalité des chemins.

222 - Instruction d'entrée

Son rôle est d'affecter à des variables les informations lues sur le support sur lequel elles ont été enregistrées, dans l'ordre de la liste d'entrée.

Syntaxe :

"instruction d'entrée" : : = lire "support d'enregistrement" : "liste d'informations à lire"

"support d'enregistrement" : : = "expression 1, entière"

"information à lire" : : = "variable J"

Sémantique :

- l'expression entière définissant le support d'enregistrement indique l'unité sur laquelle on va lire les données

Si le fichier sur lequel on demande de lire les données est vide d'information, il y aura alors interruption du programme et édition à la console-opérateur d'un message demandant l'entrée des données.

- format et syntaxe des informations à lire :

  - le format de données est libre, mais doit obéir à une syntaxe

  - syntaxe :

  x les données d'entrée (à lire) sont séparées par le signe =#= ; les blancs ne comptent pas, mais deux identificateurs contigus doivent être séparés par au moins un blanc.

  x syntaxe de chacun des types d'information à lire :

    a) Valeurs numériques, chaîne, logique :

    La syntaxe est un cas particulier de la syntaxe des expressions de même type

    Exemple : 4 =#= 5 =#= TØTØ =#= VRAI
    Lire 1 : $N1, $N2, $C1, &L1

    b) Valeur de type accès :

    La syntaxe est la même que celle définie pour un chemin d'accès

    c) Valeur de type descripteur :

    On ne peut spécifier en entrée que des descripteurs interprétés (pas de mise en facteur)

Syntaxe :

"descripteur" : : = "descripteur simple D" £ "fusion de descripteur" £ "vide"

"fusion de descripteur" : : = "descripteur simple D" "descripteur D"

"descripteur simple D" : : = "crochet gauche D" "partie relation" "partie famille" : "liste d'éléments de famille-descripteur" "crochet droit D"

"crochet X, D" : : = "crochet X, D1" £ "crochet X, D2" X ∈ {gauche, droit}

"crochet gauche D1" : : = (

"crochet gauche D2" : : = ((

"crochet droit D1" : : =)

"crochet droit D2" : : =))

"partie relation" : : = "vide" £ - "nom de relation" ("liste d'arguments")

"partie famille" : : = "vide" £ . "nom de famille"

"élément de famille-descripteur" : : = "élément de famille" "descripteur"

"nom de famille" : : = "identificateur"

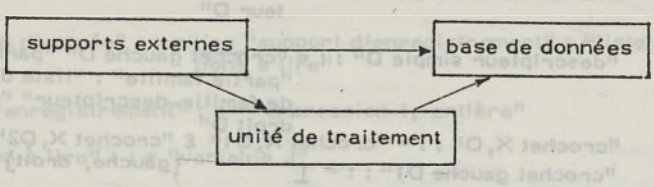
. D1 signifie descripteur élémentaire de type et

D2 signifie descripteur élémentaire de type ou

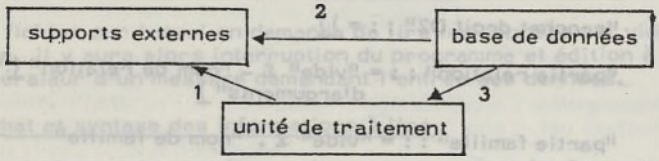
. La partie famille et la partie relation sont facultatives - l'omission de ces parties a les mêmes conséquences que celles définies dans le chapitre 1, § 32, pour les descripteurs élémentaires.

3 - RESTITUTION DES INFORMATIONS

Le paragraphe 2 traitait principalement des opérations schématisées ci-dessous :



Nous verrons dans ce chapitre les opérations dites de "restitution", transfert inverse des données par rapport aux opérations de description schématisées ci-dessous :



- Les opérations 1 et 2 pourront être effectuées grâce aux instructions de sortie

- L'opération 3 sera rendue possible grâce à l'évaluation de fonctions programmes spéciales fournies au programmeur, appelées "fonctions consultatives"

31 - Les fonctions consultatives

Une fonction consultative permet d'extraire dans le domaine des champs, des descripteurs élémentaires ou du vocabulaire un certain nombre et un certain type d'informations.

Le résultat de l'évaluation de ces fonctions seront toutes de type descripteur (J4) sauf l'une d'elles (valeur d'un noeud). Elles pourront être utilisées en tant que primaire J dans une expression J (cf. CH.2 - 41).

"fonction consultative J4" : : = "fonction consultative de descripteur" £ "fonction consultative de relation" £ "fonction consultative d'éléments liés" £ "fonction consultative des ancêtres d'un élément de base" £ "fonction consultative de niveau de noeuds" £ "fonction de listage vrai" £ "fonction de listage faux" £ "fonction d'éléments de descripteurs" £ "fonction de relation d'un descripteur" £ "fonction d'opérateurs d'un descripteur" £ "fonction de famille d'un descripteur" £ "familles d'un élément" £ "élément d'une famille"

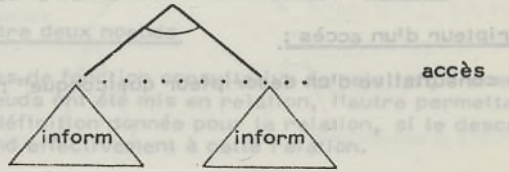
311 - Consultation des champs :

Une fonction consultative permettra, au moyen d'un accès au noeud, d'extraire dans un champ de description les informations suivantes :

nom d'un élément de base, descripteur, existence de liens, de relations, validation de relation, etc ...

L'évaluation donne toujours (sauf pour valeur d'un noeud) un descripteur plus ou moins complexe, que l'accès soit unique ou multiple.

Le 1er niveau de ce descripteur sera constitué d'un descripteur élémentaire de type et contenant les adresses d'accès auxquelles sont attachées les informations demandées (sauf pour les fonctions consultatives relatives aux relations).



- Si l'accès est unique, le descripteur élémentaire de niveau 1 ne contiendra qu'une branche.

3111 "Valeur" d'un noeud

Nous avons précédemment exposé comment accéder à un noeud dont on donne le nom.

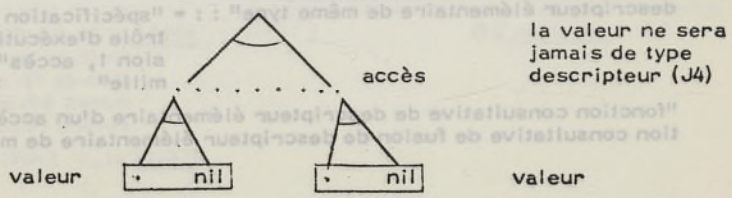
Cependant, afin d'effectuer des opérations sur les noms de ces noeuds (ou "valeurs"), il convient d'avoir la possibilité d'accéder à la valeur d'un noeud, c'est-à-dire au nom de l'élément de base porté sur ce noeud.

Syntaxe :

"fonction consultative J" : : = valeur "contrôle d'exécution" "expression 1, d'accès"

Sémantique :

. si l'accès est multiple, le résultat de l'évaluation de cette fonction sera de type descripteur (J4) sous la forme :



. pour que le résultat soit de type numérique (J1), chaîne (J2), logique (J3) ou accès (J5), il faut que l'accès soit unique

Syntaxiquement, cette fonction est à la fois de tous les types de primaire, donc peut être utilisée dans tous les types d'expression. Le contrôle de la compatibilité du type du résultat par rapport au type de l'expression dans laquelle il est utilisé ne pourra donc être effectué qu'à l'exécution.

3112 Descripteur associé à un noeud

Cette fonction consultative J4 permet d'extraire un descripteur associé à un noeud ce descripteur étant quelconque, élémentaire, fusion de descripteur élémentaire, ...

"fonction consultative de descripteur" : : = "fonction consultative d'un descripteur quelconque" £ "fonction consultative de fusion de descripteur élémentaire de même type"

a - Descripteur d'un accès :

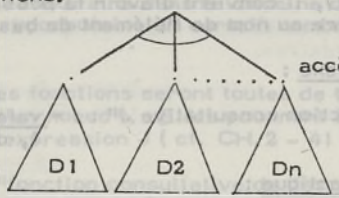
"fonction consultative d'un descripteur quelconque" : : = descripteur "contrôle d'exécution" "expression 1, accès" "spécification de profondeur"

"fonction consultative de descripteur d'un accès" : : = "fonction consultative d'un descripteur quelconque"

"spécification de profondeur" : : = "vide" £ jusqu'à "expression 1, entière"

Résultat de l'évaluation :

Le descripteur à un ou plusieurs niveaux (profondeur, spécifiée ou non) attaché à chacun des éléments de base auquel on accède, interprétation faite de mises en facteur et des liens.



b - Fusion de descripteurs élémentaires

"fonction consultative de fusion de descripteur élémentaire de même type" : : = "spécification de type" "contrôle d'exécution" "expression 1, accès" "relation famille"

"fonction consultative de descripteur élémentaire d'un accès" : : = "fonction consultative de fusion de descripteur élémentaire de même type"

"spécification de type" : : = composants £ alternatives

"relation famille" : : = "relation" "famille"

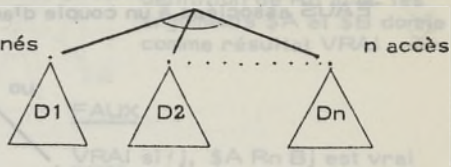
"relation" : : = "vide" £ relation "liste d'indicateur de relation"

"famille" : : = "vide" £ famille "liste d'identificateur de famille"

Résultat de l'évaluation :

L'ensemble des descripteurs élémentaires fusionnés de type spécifié (composants : type et, alternatives : type ou) et de famille et de relations spécifiées ou non.

descripteurs élémentaires fusionnés de même type



3113 Relations entre deux noeuds

Il existe deux types de fonction consultative de relation, l'une permettant de savoir si deux noeuds ont été mis en relation, l'autre permettant de savoir, en fonction de la définition donnée pour la relation, si le descripteur des deux noeuds répond effectivement à cette relation.

"fonction consultative de relation" : : = "f. c. d'éléments mis en relation" £
"f. c. de validation d'une relation"

a - Test sur l'existence de relation entre deux noeuds

"f. c. d'éléments mis en relation" : : = ? "indicateur de relation"

. évaluation et exemples :

1) La relation n'est pas spécifiée :

? relation $A, $B donne comme résultat VRAI si ∃Ri | Ri $A, $B
FAUX si ∄Ri | Ri $A, $B

? relation $A, nil VRAI si ∃Rs, $Bj | Ri $A, $Bj
FAUX si ∄Ri, $Bj | Ri $A, $Bj

? relation nil, $B analogue

? relation nil, nil

2) La relation est spécifiée

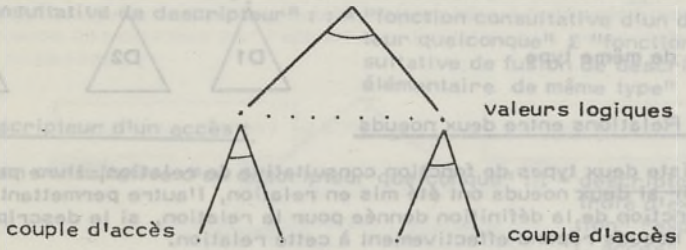
? Rn $A, $B — VRAI / FAUX

? Rn $A, nil — VRAI si ∃$Bj Rn $A, $Bj / FAUX si ∄$Bj Rn $A, $Bj

? Rn nil, $B — analogue

? Rn nil, nil — ———

L'évaluation donne toujours une liste de valeurs logiques VRAI ou FAUS associées à un couple d'accès sous la forme :



- si l'on veut connaître les éléments (relation ou accès) qui rendent cette fonction VRAIE ou FAUSSE, on utilisera une autre fonction (cf. 3114)

b - Test de validité d'une relation entre deux noeuds :

. l'évaluation de cette fonction active le bloc de définition de la relation

"f. c. de validation d'une relation" : : = "contrôle d'exécution" "expression 1 d'accès" "indicateur de relation" "contrôle d'exécution" "expression d'accès"

"indication de relation" : : = ? "identificateur de relation" £ ? "identification de relation" "liste d'arguments"

. évaluation et exemples :

1) La relation n'est pas spécifiée :

$A ? relation $B — VRAI si ∃i, $A Ri $B VRAI (l'activation du bloc de définition de Ri avec les arguments $A et $B donne comme résultat VRAI)

FAUX si ∄i $ARi $B FAUX

$A ? relation nil — VRAI si ∀i, A Ri Bj VRAI
FAUX si ∃i ou ∃j A.Ri.Bj FAUX

nil ? relation $B — analogue

nil ? relation nil — analogue

2) La relation est spécifiée

$A ? Rn $B — VRAI si l'évaluation du bloc de définition de Rn avec les arguments $A et $B donne comme résultat VRAI

ou

FAUX

$A ? Rn nil — VRAI si ∀j, $A Rn Bj est vrai
FAUX si ∃j $A Rn Bj est faux

nil ? Rn $B — analogue

nil ? Rn nil — analogue

L'évaluation donnc toujours un résultat ayant la même forme qu'en a.

c - Test sur l'existence de liens entre 2 noeuds

Le lien est une relation particulière dont on ne consultera que l'existence puisqu'elle est toujours valide

"fonction consultative d'éléments liés" : : = ? lien "contrôle d'exécution" "expression 1, accès", "contrôle d'exécution" "expression 1, accès"

L'évaluation de cette fonction est identique à l'évaluation d'une fonction consultative d'éléments mis en relation dans laquelle la relation est spécifiée.

3114 - Fonctions de listage associées au résultat d'une fonction consultative de relation :

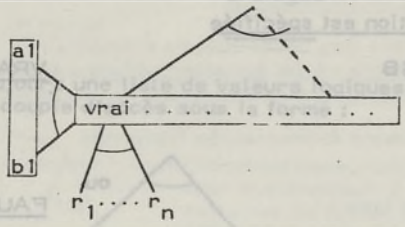
Ces fonctions permettent de connaître les éléments (relations ou/et accès) qui rendent une fonction consultative de relation vraie ou fausse

"fonction de listage vrai" : : = vrai ("fonction consultative de relation")
£ vrai ("fonction consultative d'éléments liés")

Evaluation :

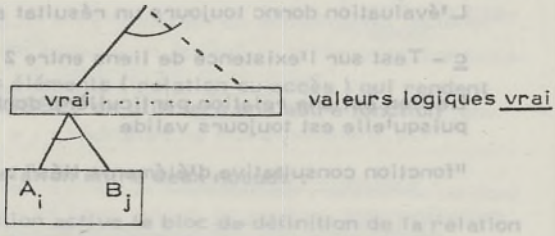
Le résultat sera l'ensemble des éléments qui rendent la fonction de consultation de relation vraie

. dans les cas a (1), b (1) : (relation inconnue)



. dans les cas a(2, b(3) et c : (relation spécifiée)

c'est une extraction du descripteur-résultat de l'évaluation de la fonction consultative, des descripteurs associés aux valeurs logiques vrai



"fonction de listage faux" : : = faux ("fonction consultative de validation d'une relation")

Evaluation :

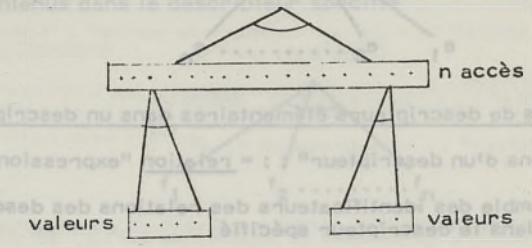
Cette fonction ne sera utilisée que dans le cas d'une validation de relation, puisque son résultat est toujours le descripteur vide dans les autres cas. La structure du descripteur-résultat est identique à celles spécifiées précédemment pour la fonction de listage vrai.

3115 - Ancêtres d'un élément de base :

"f.c. des ancêtres d'un élément de base" : : = anc "contrôle d'exécution" "expression 1, accès"

Evaluation :

Liste des valeurs des noeuds situés sur le chemin d'accès au noeud :

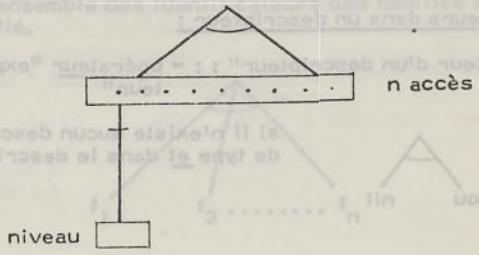


3116 - Niveau d'un noeud :

"f.c. de niveau de noeuds" : : = niveau "contrôle d'exécution" "expression 1, accès"

Evaluation :

Une valeur numérique entière représentant le niveau du noeud associée à chacun des noeuds atteints



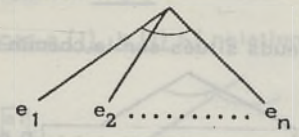
312 - Consultation de descripteurs :

3121 - Eléments d'un descripteur :

Cette fonction donne comme résultat la liste des valeurs des éléments de base d'un descripteur quelconque.

"f.c. d'éléments de descripteurs" : : = élément "expression 1, descripteur"

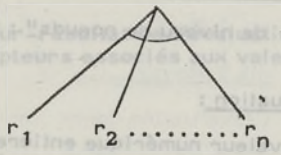
Résultat : ensemble de valeurs numériques, chaîne, logique ou accès sous la forme d'un descripteur élémentaire de type et :



3122 - Relations de descripteurs élémentaires dans un descripteur :

"f.c. de relations d'un descripteur" : : = relation "expression 1, descripteur"

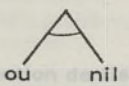
Résultat : ensemble des identificateurs des relations des descripteurs élémentaires obtenus dans le descripteur spécifié



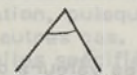
3123 - Opérateurs dans un descripteur :

"f.c. d'opérateur d'un descripteur" : : = opérateur "expression 1, descripteur"

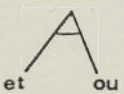
Résultat :



si il n'existe aucun descripteur élémentaire de type et dans le descripteur spécifié



si il n'existe aucun descripteur élémentaire de type ou dans le descripteur spécifié

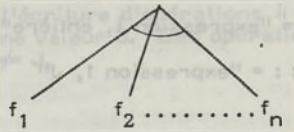


si le descripteur est hétérogène vis-à-vis du type de descripteur élémentaire qu'il contient

3124 - Familles d'un descripteur

"f.c. de familles d'un descripteur" : : = famille "expression 1, descripteur"

Résultat : ensemble des identificateurs de famille des descripteurs élémentaires contenus dans le descripteur spécifié



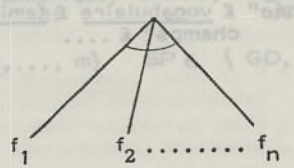
313 - Consultation du vocabulaire :

3131 - Familles d'un élément du vocabulaire :

"famille d'un élément" : : = famille "spécification de l'élément"

"spécification de l'élément" : : = "expression 1, arithmétique" £ "expression 1, chaîne" £ "expression 1, logique" £ "expression 1, accès"

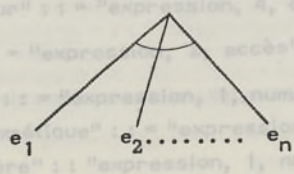
• évaluation : ensemble des identificateurs des familles auxquelles appartient l'élément spécifié.



3132 - Eléments d'une famille :

"élément d'une famille" : : = felem "identificateur de famille"

• évaluation : ensemble des éléments de la famille spécifiée



32 - Instruction de sortie :

Son rôle est d'éditer, sur un support choisi par le programmeur, un certain nombre d'informations.

Syntaxe :

"instruction de sortie" : : = éditer "support de sortie" : "liste d'informations à éditer"

"support de sortie" : : = "expression 1, entière"

"information à éditer" : : = "expression 1, J"

Sémantique :

- l'expression entière définissant le support de sortie, définira également la forme de cette sortie en fonction du support et des desiderata du programmeur : par exemple, la sortie d'un plan sous la forme de dessin (et non sous forme d'arbre) sur un écran de visualisation sera relatif à un code numérique n donné.

- éditer un accès, c'est éditer l'énumération des noeuds du chemin d'accès.

- on peut demander l'édition de toute information contenue dans une base de données, y compris la base de données toute entière; on utilisera une fonction consultative de descripteur d'un accès, l'accès étant un accès par nom, pris dans la liste suivante :

"nom de la base de donnée" £ vocabulaire £ famille £ champs £ "nom de champs" £ ....

4 - OPERATIONS DE TRAITEMENT

Elles consistent, à partir de données contenues dans l'unité de traitement, à générer d'autres données de même type ou de types différents.

41 - Expressions et définition d'opérateurs et de comparateurs

411 - Expressions

Une expression J est l'écriture d'opérations à effectuer dans un certain ordre afin d'obtenir une valeur J. Les opérations peuvent porter sur des expressions I avec I $\neq$ J.

Exemple :

La comparaison de deux entiers donne une valeur logique.

Le résultat d'une fonction programme peut être obtenu après de longs calculs complexes.

Dans les règles de grammaire qui suivent, écrire :

"A, J" : : = "B, J" : = "C, J" avec J $\in$ { entier, logique }

revient à écrire :

"A, entier" : : = "B, entier" : = "C, entier"
"A, logique" : : = "B, logique" : = "C, logique"

Dans ce qui suit $J \in \{\underset{J1}{\text{numérique}}, \underset{J2}{\text{chaîne}}, \underset{J3}{\text{logique}}, \underset{J4}{\text{descripteur}}, \underset{J5}{\text{accès}}\}$

$n \in \{1, \ldots, m\}$ $SP \in \{ GD, DG \}$ sens de parenthésage

GD : (x # x) # x
DG : x # (x # x)
G : x #
D : # x

"primaire, numérique" : : = "expression, 4, numérique"

"primaire, logique" : : = "expression, 4, logique"

"primaire, chaîne" : : = "expression, 2, chaîne"

"primaire, descripteur" : : = "expression, 4, descripteur"

"primaire, accès" : : = "expression, 2, accès"

"expression entière" : : = "expression, 1, numérique"

"expression, 1, arithmétique" : : = "expression, 1, numérique"

"expression, 1, entière" : : "expression, 1, numérique"

Une expression entière est une expression numérique dont le résultat est un entier, sinon il y a troncature.

"expression, n, J" : : = "expression, n + 1, J" opérateur, SP, n, J "expression, n, J" £ "opérateur, G, n, J" "expression, n + 1, J" £ "expression, n + 1, J" "opérateur, D, n, J" £ "expression, n + 1, J"

4111 - Expressions numériques

"expression, 4, numérique" : : = "variable, numérique " £ "nombre" £ nil £ "indicateur de fonction numérique" £ ("expression, 1, numérique")

"opérateur, M, 1, numérique" : : = + £ - £ "indicateur d'opérateur, M, 1, numérique"

M ∈ {GD, G}

"opérateur, GD, 2, numérique" : : = * £ / £ "indicateur d'opérateur, GD, 2, numérique"

"opérateur, DG, 3, numérique" : : = ** £ "indicateur d'opérateur, DG, 3, numérique"

"nombre" : : = "entier" £ . "entier" £ "entier" . £ "entier" . "entier"

"entier" : : = "chiffre" £ "chiffre" "entier"

"indicateur de fonction, numérique" : : = "fonction consultative, numérique" £ "fonction programme, numérique"

"nombre entier" : : = "entier"

4112 - Expressions logiques

"expression, 4, logique" : : = "variable, logique" £ "valeur logique" £ nil £ "indicateur de fonction, logique" £ ("expression, 1, logique") £ "expression, 1, J" "opérateur de comparaison, J" "expression, 1, J"

J ∈ {numérique, chaîne}

"opérateur de comparaison, J" : : = = £ ≠ £ > £ < £ >= £ <= £ "indicateur d'opérateur de comparaison, J"

"opérateur de comparaison, logique" : : = équiv £ "indicateur d'opérateur de comparaison, logique"

"opérateur de comparaison, descripteur" : : = analogue £ identique £ "indicateur d'opérateur de comparaison, descripteur"

La syntaxe impose que tous les opérateurs de même priorité aient le même sens de parenthésage.

- Deux descripteurs sont identiques s'ils se "recouvrent". L'ordre des éléments de base dans chaque descripteur élémentaire ou à 1 niveau compte.

- Si deux descripteurs sont égaux à l'ordre près des éléments à chaque niveau, on dit qu'ils sont analogues.

"opérateur de comparaison, accès" : : = = £ #

"opérateur, GD, 1, logique" : : = ou £ "indicateur d'opérateur,GD, 1, logique"

"opérateur, GD, 2, logique" : : = et £ "indicateur d'opérateur GD, 2, logique"

"opérateur, G, 3, logique" : : = non £ "indicateur d'opérateur, G, 3, logique"

"valeur logique" : : = vrai £ faux

"indicateur de fonction, logique" : : = "fonction consultation, logique" £ "fonction programme, logique"

4113 - Expressions chaîne

"expression, 2, chaîne" : : = " valeur chaîne" £ ("expression, 1 chaîne") £ "variable, chaîne" £ "indicateur de fonction, chaîne"

"valeur chaîne" : : = ! "suite de caractères" ! £ nil £ "identificateur"

"indicateur de fonction, chaîne" : : = "fonction consultative, chaîne" £ "fonction programme, chaîne"

"caractère" : : = "lettre" £ "chiffre" £ . £ ? £ ; £ (£) £ / £ = £ # £ > £ < £ * £ _ £ / £ + £ - £ . £ " £ ... etc

"opérateur, GD, 1, chaîne" : : = conc £ "indicateur d'opérateur, GD, 1, chaîne"

4114 - Expression de descripteur

"expression, 4, descripteur" : : = "variable descripteur" £ "descripteur" £ "indicateur de fonction, descripteur" £ nil £ ("expression, 1, descripteur ")

"opérateur, GD, 1, descripteur" : : = union £ "indicateur d'opérateur, GD, 1, descripteur

"opérateur, GD, 2, descripteur" : : = inter £ "indicateur d'opérateur, GD, 2, descripteur"

"opérateur, G, 3, descripteur" : : = supphc "liste d'éléments de famille" : £ rajouthc "liste d'éléments de famille" :

Restriction : ces opérateurs ne s'appliquent qu'à une expression de type descripteur dont la valeur est un descripteur élémentaire (supphc et rajouthc)

"indicateur de fonction, descripteur" : : = "fonction consultative" £ "fonction programme, descripteur"

4115 - Expression accès

"expression, 2, accès" : : = "variable accès" £ "indicateur de fonction, accès" £ ("expression, 1, accès") £ nil £ "chemin d'accès"

"chemin d'accès" : : = "chemin d'accès unique" £ "chemin d'accès multiple"

"opérateur, D, 1, accès" : : = ("expression, 1, numérique", "expression, 1, numérique")

Restriction : cet opérateur ne s'applique qu'à une expression d'accès dont la valeur est un accès unique. Le résultat de l'évaluation des expressions entre parenthèses doit être entier, sinon il y aura troncature.

Sémantique :

La première expression indique la valeur relative du déplacement vertical

+ n : descendre de n niveaux
- n : monter de n niveaux

La deuxième expression indique la valeur relative du déplacement horizontal

+ n ; + : à droite
- n ; - : à gauche n : nombre d'éléments de base

Le résultat est l'accès trouvé après avoir effectué les déplacements dans l'ordre à partir de l'accès spécifié par l'évaluation de l'expression à laquelle s'applique cet opérateur.

"indicateur de fonction, accès" : : = "fonction consultative, accès" £ "fonction programme, accès"
"fonction programme, J" : : = "identificateur" "argument p"

412 - Définition d'opérateurs et de comparateurs

Les définitions d'opérateurs et de comparateurs sont des moyens laissés à l'architecte pour se libérer de la contrainte des opérateurs et comparateurs fournis avec le langage (+ , - , * , / , > , < , > = , < = , = , # , équiv, analogue, identique) et de créer ceux dont il a réellement besoin.

"tête de bloc" : : = ... £ opérateur "tête d'opérateur" £ comparer "tête de comparaison"

4121 - Définition d'opérateurs

"tête d'opérateur" : : = "nom d'opérateur" "propriétés" ; ("argop")

"nom d'opérateur" : : = "identificateur"

"propriétés" : : = priorité "expression entière", "type", "sens de parenthésage"

"sens de parenthésage" : : = x # (x #) £ (x # x) # x £ # x £ #

"argop" : : = "argument" ; "argument" £ "argument"

4122 - Définition de comparateurs

"tête de comparaison" : : = "nom de comparateur" ("argument", "argument")

"nom de comparateur" : : = "identificateur"

"indicateur d'opérateur, SP, N, J" : : = "identificateur"

SP ∈ { GD, DG, G, D }

N ∈ N

J ∈ { numérique, logique, chaîne, accès, descripteur }

115 - Expression accès

"expression, 2, accès" : : = "variable accès" £ "indicateur de fonction, accès" £ ("expression, 1, accès") £ nil £ "chemin d'accès"

"chemin d'accès" : : = "chemin d'accès unique" £ "chemin d'accès multiple"

"opérateur, D, 1, accès" : : = ("expression, 1, numérique", "expression, 1, numérique")

Restriction : cet opérateur ne s'applique qu'à une expression d'accès dont la valeur est un accès unique. Le résultat de l'évaluation des expressions entre parenthèses doit être entier, sinon il y aura troncature.

Sémantique :

La première expression indique la valeur relative du déplacement vertical

+n : descendre de n niveaux
-n : monter de n niveaux

La deuxième expression indique la valeur relative du déplacement horizontal

+n ; + : à droite
      - : à gauche n : nombre d'éléments de base

Le résultat est l'accès trouvé après avoir effectué les déplacements dans l'ordre à partir de l'accès spécifié par l'évaluation de l'expression à laquelle s'applique cet opérateur.

"indicateur de fonction, accès" : : = "fonction consultative, accès" £ "fonction programme, accès"
"fonction programme, J" : : = "identificateur" "argument p"

12 - Définition d'opérateurs et de comparateurs

Les définitions d'opérateurs et de comparateurs sont des moyens laissés à l'architecte pour se libérer de la contrainte des opérateurs et comparateurs fournis avec le langage (+ , - , * , / , > , < , >= , <= , = , # , équiv, analogue, identique) et de créer ceux dont il a réellement besoin.

"tête de bloc" : : = ... £ opérateur "tête d'opérateur" £ comparer "tête de comparaison"

121 - Définition d'opérateurs

"tête d'opérateur" : : = "nom d'opérateur" "propriétés" ; ("argop")

"nom d'opérateur" : : = "identificateur"

"propriétés" : : = priorité "expression entière", "type", "sens de parenthésage"

4221 Transformation d'un nombre en chaîne et vice-versa

a) fonction TRNC : passage d'un nombre à une chaîne

Syntaxe :

TRNC ("expression numérique")

Le résultat de l'exécution de cette fonction donnera une chaîne :

Ex : TRNC (3638 + (54 / 2)) ⟶ ! 3665 !

b) fonction TRCN : passage d'une chaîne à un nombre

Syntaxe :

TRCN ("expression chaîne")

L'expression ne doit contenir que des caractères numériques et les caractères + , - , .

Ex : TRCN (! 45.7 !) ⟶ 45.7

4222 Transformation d'une chaîne en descripteur et vice-versa

a) fonction TRCD : passage d'une chaîne à un descripteur

Syntaxe : TRCD ("expression chaîne")

Le résultat de l'exécution donnera un descripteur élémentaire de type et

Ex : TRCD (! SALLE_DE_COURS !) —— SALLE_DE_COURS

b) fonction TRDC : passage d'un descripteur élémentaire de type et dont les éléments ne sont que des caractères à une chaîne

TRDC ("expression J4 élémentaire de type et")

"expression, J4, élémentaire de type et" : : = "expression, 1, descripteur"

43 - Les fonctions de description

En ARLANG, nous avons introduit un type de fonction particulier, appelé "fonction de description"

L'activation d'une telle fonction en cours de programme donne comme résultat un descripteur élémentaire.

431 - Bloc de définition d'une fonction de description :

Ce bloc, analogue aux autres blocs de programme, a pour but de définir l'ensemble des instructions à exécuter pour générer les valeurs du descripteur

élémentaire dans lequel il est situé (cf. § 432) et cela en fonction du noeud auquel ce descripteur est attaché.

Syntaxe :

"tête de fonction de description" : : = fonction "reste de tête de fonction de description"

"reste de tête de fonction de description" : : = "identificateur" "argument d'accès" "argument de relation" "argument de famille" "argument p"

"argument d'accès" : : = "contrôle d'exécution" "expression, 1, accès"

"argument de relation" : : = "spécification de relation"

"argument de famille" : : = "spécification de famille"

Sémantique :

- L'argument d'accès représentera le noeud auquel sera attaché le descripteur élémentaire, l'argument de relation la relation du descripteur élémentaire (si la relation n'est pas spécifiée, le descripteur est de type et), l'argument de famille est la famille spécifiée dans l'expression de descripteur

- L'exécution de l'instruction composée doit fournir au moins un descripteur élémentaire dont les éléments sont de type numérique, logique, chaîne ou accès et homogène.

Ex : fonction SURFACE $A

début

----
fin

432 - Mise en place d'une fonction de description dans une arborescence :

Syntaxe

Rappel :

"descripteur élémentaire" : : = "spécification de relation" "spécification de famille" : "spécification de sélection des éléments du descripteur"

"spécification de sélection des éléments du descripteur" : : = "énumération" £ "spécification de fonction de description"

"spécification de fonction de description" : : = "indicateur de fonction de description" "spécification d'activation"

"indicateur de fonction de description" : : " "nom de fonction" "argument"

"spécification d'activation" : : = unique £ multiple

Sémantique

- Les éléments du descripteur élémentaire seront le résultat de l'activation (cf. 433) de la fonction de description.

- Le paramètre effectif mis en lieu et place de l'argument d'accès sera le noeud auquel est attaché le descripteur élémentaire, les arguments de relation et de famille également.

Exemple :

attach $A; - SURF . N : SURFACE ;

433 - Activation d'une fonction de description

L'activation d'une fonction peut être effectuée explicitement ou implicitement. Il conviendra qu'au moment de l'activation, le programmeur s'assure de la possibilité d'évaluation de la fonction.

4331 - Activation explicite :

Syntaxe :

"appel de fonction de description" : : = activer "tête de fonction de description"

Sémantique :

Si au moment de la création du descripteur élémentaire évalué par la fonction de description,

1) on a spécifié une activation unique, alors on ne pourra pas re-demander une nouvelle activation de cette fonction à cet accès; le descripteur calculé par la première et unique activation prend la place de la fonction de description.

2) si on a spécifié une activation multiple, on pourra alors re-demander une nouvelle activation de la fonction en cet accès, le résultat de cette nouvelle évaluation se substituera à l'ancien résultat jusqu'à nouvelle activation.

4332 - Activation implicite :

Chaque fois qu'un accès (où qu'il apparaisse) utilise au moins un élément de base généré par la fonction de description, il déclenche une activation.

Les accès peuvent être utilisés de manière indirecte (par exemple si l'on utilise une fonction consultative de descripteur).

Dans tous les cas on ne pourra pas citer nomément le noeud auquel on accède puisqu'il est généré après coup. En pratique, c'est l'expression d'accès multiple qui déclenche cette activation; ex. valeur $A. TØTØ :*

- Dans le cas d'une activation unique la fonction n'est activée qu'une seule fois

- Dans le cas d'une activation multiple la fonction est re-calculée à chaque fois qu'on accède aux valeurs créées par l'appel précédent. Cela a pour effet que les éléments de descripteur de l'appel précédent sont substitués par les éléments de descripteur du nouvel appel. Les éléments de descripteur de l'appel précédent sont perdus ainsi que toutes les modifications qui auraient pu les affecter. Dans le cas où le programmeur veut éviter cet effet il peut écrire :

$D : = comp (A) famil TOTO; (1er appel)
détach comp (A) famil TOTO; (2me appel et détachement)
attach A, $D;

Cette suite d'instructions permet de rendre unique à un moment donné une activation déclarée multiple.

5 - OPERATIONS DE CONTROLE D'ALGORITHME

Par "contrôle d'algorithme", nous entendons :

- répétition d'opérations
- exécution d'opérations sous condition
- branchement
- activation de programme
- interruption de programme (mode interactif)

51 - Répétition d'opérations :

Elle a pour rôle de répéter n fois une chaîne d'instructions en faisant varier la valeur d'une variable ou en testant à chaque passage la valeur d'une expression logique.

Syntaxe :

"instruction itérative" : : = "instruction pour" £ "instruction tant que"

"instruction pour" : : = pour "variable numérique" "corps d'instruction pour" faire "instruction"

"corps d'instruction pour" : : = allant de "borne inférieure" "incrément-borne supérieure" £ : = "liste d'expression 1, numérique"

"incrément-borne supérieure" : : = "borne supérieure" £ incrément "expression 1, entière" "borne supérieure"

"borne supérieure" : : = jusqu'à "expression 1, numérique"

"borne inférieure" : : = "expression 1, numérique"

"instruction tant que" : : = tant que "expression 1, logique" faire "instruction"

52 - Exécution d'opérations sous condition :

Syntaxe :

"instruction conditionnelle" : : = si "expression 1, logique" " reste d'instruction conditionnelle"

"reste d'instruction conditionnelle" : : = faire "instruction" £ alors "instruction" sinon "instruction"

53 - Opérations de branchement :

Syntaxe :

"instruction de branchement" : : = "instruction allera" £ "instruction choix"

531 - Instruction allera :

"instruction allera" : : = allera "étiquette"

"étiquette" : : = "identificateur"

Sémantique :

Une instruction allera permet de se brancher, en un point de l'exécution du programme, sur une instruction non consécutive à la dernière exécutée, et repérée par une étiquette.

"instruction étiquetée" : : = "étiquette" : "instruction"

Si la référence se trouve en dehors du bloc où apparaît l'instruction allera, cette étiquette doit être déclarée en global au même titre que les autres identificateurs :

"étiquette globale" : : = global "liste d'étiquettes"

532 - Instruction choix :

"instruction choix" : : = choix "expression 1, entière" parmi début "chaîne d'instructions" fin

Sémantique :

- L'évaluation de l'expression entière donne un nombre qui est le rang de l'instruction qui doit être exécutée dans la chaîne spécifiée.

- Si l'expression entière est inférieure ou égale à zéro ou supérieure au nombre d'instructions de la chaîne, un message sera émis et l'instruction choix sera sautée.

54 - Activation de programme :

Activer un programme consiste à faire appel, dans un programme, à un bloc de définition de programme exécutable défini indépendamment et enregistré dans la base de programmes.

Syntaxe :

"appel de programme" : : = activer "identificateur" "argument p"

Sémantique :

- La récursivité est permise en ARLANG, ainsi que le non-déterminisme par l'affectation d'identificateurs identiques à des blocs de programme différents. L'activation de plusieurs de ces blocs se fera alors dans l'ordre où ils ont été créés; leur sélection dépendra des "patterns" placés en argument de ces programmes.

55 - Interruption de programme :

Les instructions d'interruption permettent d'écrire des programmes interactifs, c'est-à-dire permettant le contrôle de l'exécution du programme par l'opérateur.

"instruction d'interruption" : : = "instruction PAUSE" £ "instruction STOP"

"instruction pause" : : = pause "expression 1, chaîne"

"instruction stop" : : = stop "expression 1, chaîne"

Sémantique :

- L'instruction stop exécutée assure la fin de la session, "rend la main" au système et édite le message contenu dans l'expression chaîne.

- L'instruction pause assure un arrêt momentané de la session en cours avec édition du message. La main est donnée au programmeur = il peut alors changer un disque, une bande, passer en mode bureau, ...

Le temps d'inaction pendant la pause est limité. Au-delà de cette limite, la pause est transformée en STOP.

La reprise se fait par une instruction de reprise :
"instruction de reprise" : : = reprendre

L'exécution de la session en cours, momentanément interrompue, est alors reprise à l'instruction suivant l'instruction PAUSE.

53 - Opérations de branchement :

Syntaxe :

Sémantique :

531 - Instruction allera :

Sémantique :

Une instruction allera permet de se brancher, dans un programme, sur une instruction non consécutive à la dernière exécutée, et active, c'est-à-dire permettant le contrôle de l'exécution du programme par l'opérateur.

Si la référence se trouve en dehors du bloc du programme en cours, cette étiquette doit être globale.

Syntaxe :

Sémantique :

L'évaluation de l'expression entière donne le numéro de l'instruction qui doit être exécutée dans la chaîne spécifiée.

Si l'expression entière est supérieure au nombre d'instructions de la chaîne spécifiée, l'instruction choix sera sautée.

54 - Activation de programme :

Activer un programme consiste à faire appel, dans un programme, à un bloc de définition de programme exécutable défini indépendamment et enregistré dans la base de programmes.

PARTIE III : Scénarios d'utilisation du langage en conception assistée

L'objet de cette troisième partie est de présenter, au travers de quelques scénarios pris dans le contexte du processus de conception architecturale, quelle peut être l'utilisation du langage ARLANG et ses apports relativement à l'usage des procédures manuelles ou automatisées.

1 - INTRODUCTION

Les scénarios contenus dans le présent document ont pour objet : d'une part de situer par rapport à la pratique professionnelle la recherche que nous avons menée jusqu'à ce jour, d'autre part d'illustrer l'utilisation du langage par des exemples puisés parmi les problèmes concrets qui se posent à l'architecte.

Ces exemples permettront au lecteur de se familiariser avec les notions du langage exposées dans la partie II.

L'assimilation des règles d'utilisation du langage repose sur la compréhension de la notion d' "état de base de données" et d' "évolution d'état de base de données".

Favoriser la compréhension de ces notions est le premier objectif que nous avons visé dans l'élaboration de ces scénarios.

Les bases de données correspondant à une réalité, il nous fallait indiquer par rapport à quelle réalité se situaient les bases de données que nous allions décrire; d'où la nécessité de présenter le champ plus général sur lequel la plupart des bases de données ARLANG vont porter.

Par ailleurs l'utilisation du langage ne serait pas possible si elle ne s'appuyait sur des démarches explicites, d'où le troisième aspect de ces scénarios : l'aspect méthodologique au travers duquel nous montrons l'effort de rationalisation que devront réaliser les futurs utilisateurs du langage.

De ces trois considérations découle la forme que nous avons donnée à ces scénarios, c'est-à-dire :

- Etude d'un scénario globalisant le processus de conception architecturale tel que nous avons pu constater son déroulement tant sous sa forme courante que sous sa forme partiellement informatisée,

- Ensemble de scénarios ARLANG situés par rapport au scénario global pour montrer :

  - l'évolution de l'état des bases de données ARLANG,

  - la formalisation ARLANG des différentes phases du processus de conception architecturale,

  - la formalisation d'opérations diverses prises à titre d'exemple dans les différentes phases du processus,

  - la formalisation d'exemples de description dans les bases de données ARLANG.

2 - SCENARIO TRAITANT DU PROCESSUS DE CONCEPTION ARCHITECTURALE DANS SON ENSEMBLE

21 - Scénario

Ce scénario a pour objet de formaliser un modèle de processus de conception architecturale. Nous avons pensé qu'il était nécessaire de disposer d'un modèle de ce type afin de pouvoir situer la place qu'occupe par rapport à ce modèle, ou dans ce modèle, chacun des autres scénarios que nous présentons.

Le processus que nous présentons dans ce scénario est très proche de ceux qui découlent des méthodes précédemment élaborées au GAMSAU (notamment Annexe 1).

Toutes les étapes, feed-backs et options possibles ne sont pas mentionnés mais l'idée de processus de conception architecturale que nous avons dégagée est présente; ce scénario est aussi pour nous l'occasion de définir certains concepts, de préciser la place et la définition de certains termes dans la description du processus de conception architecturale. Ainsi, l'ensemble scénario et définitions qui s'y attachent représentent pour nous les bases (1) d'une discussion sur l'utilisation du langage . Il nous paraît essentiel pour mener à bien notre travail de mettre en oeuvre tous les moyens qui permettront au lecteur de ne pas avoir l'impression de se trouver en face de textes ésotériques; pour cela il nous faut préciser la plupart des termes qui pourraient être l'objet de définition polysémique ou dont la définition ne serait pas connue du lecteur.

Or, l'élaboration du cahier des charges du langage ARLANG et la rédaction des scénarios utilisant ce langage nous ont montré à quel point l'imprécision de la terminologie qu'utilisent les intervenants du processus de conception architecturale était source d'erreur et d'incompréhension. Nous pensons qu'une des qualités d'ARLANG sera de stimuler (2) les intervenants à produire un effort dans la définition de leur langage, cet effort leur permettant de bénéficier des avantages (2) que pourra procurer l'utilisation d'ARLANG.

(1) Bien que ce modèle de processus ait servi d'argument à la recherche des objectifs de notre travail, celui-ci n'est en fait qu'une méthode parmi d'autres auxquelles le langage pourra tout aussi bien s'appliquer.

(2) Notamment : faciliter l'élaboration de concepts, permettre une modification aisée et rapide des documents véhiculant l'information.

22 - Lexique du scénario général

2.21 Généralités

1 - Processus de conception architecturale (PCA)

C'est la production de connaissances développée à partir (1) de l'analyse d'un besoin ainsi que la description et la mise en oeuvre des moyens permettant de satisfaire ce besoin en termes d'aménagements architecturaux.

2 - Besoin général

Besoin, exprimé ou non, que produit "l'espace urbain" (2) à un moment donné.

3 - Besoin particulier

Besoin résultant de la confrontation du besoin général aux contraintes des conditions particulières (3) sous lesquelles on veut que soit satisfait le besoin général.

4 - Aléa - Maîtrise de l'aléa

Le processus de conception architecturale est aléatoire, ceci est une notion fondamentale qui condamne toute idée de conception automatique de l'architecture parce que le facteur temps a une incidence considérable (4)

6 - A.M.O. - Accord du maître d'oeuvre

(1) Nous considérons que le processus de conception architecturale commence avec la constitution du premier groupe de réflexion sur le besoin général même si ce groupe ne comporte pas d'architectes parmi ses membres.

(2) Espace bâti + espace social + espace défini à partir des flux qui les anim

(3) Le terme de condition particulière signifie que l'on va satisfaire le beso général seulement dans le cadre de certains éléments, de l'espace urbai considérés comme déterminants.

(4) (Entre le moment où l'on prend la décision d'aménager et celui où l'on r lise il se produit des évènements imprévisibles)

2.22 Phases générales du processus de conception architecturale

Dans ce scénario nous avons essayé de découper le processus de conception architecturale en trois phases générales; il va de soi que ce découpage n'a pour but que de faciliter la compréhension du processus. En réalité, bien que ces phases générales existent, la définition des articulations reste difficile, car le dévaloppement du processus n'est pas linéaire mais itératif.

(10) Phase d'intention : c'est la phase au cours de laquelle le maître de l'ouvrage prend en compte un "besoin général" dans l'intention de le satisfaire sous certaines conditions.

(11) Phase de conception

C'est la phase au cours de laquelle le maître d'oeuvre exécute ou fait exécuter les recherches et la description des aménagements architecturaux qui satisferont les objectifs déterminés dans la phase d'intention.

(12) Phase de conception théorique: phase au cours de laquelle on va analyser les objectifs déterminés au cours de la phase d'intention pour en tirer un programme d'organisation spatiale ou définir un parti architectural et son synopsis.

(18) (13) Phase d'instrumentation : phase au cours de laquelle on va rechercher, étudier et décrire les solutions techniques qui seront utilisées pour satisfaire le programme architectural.

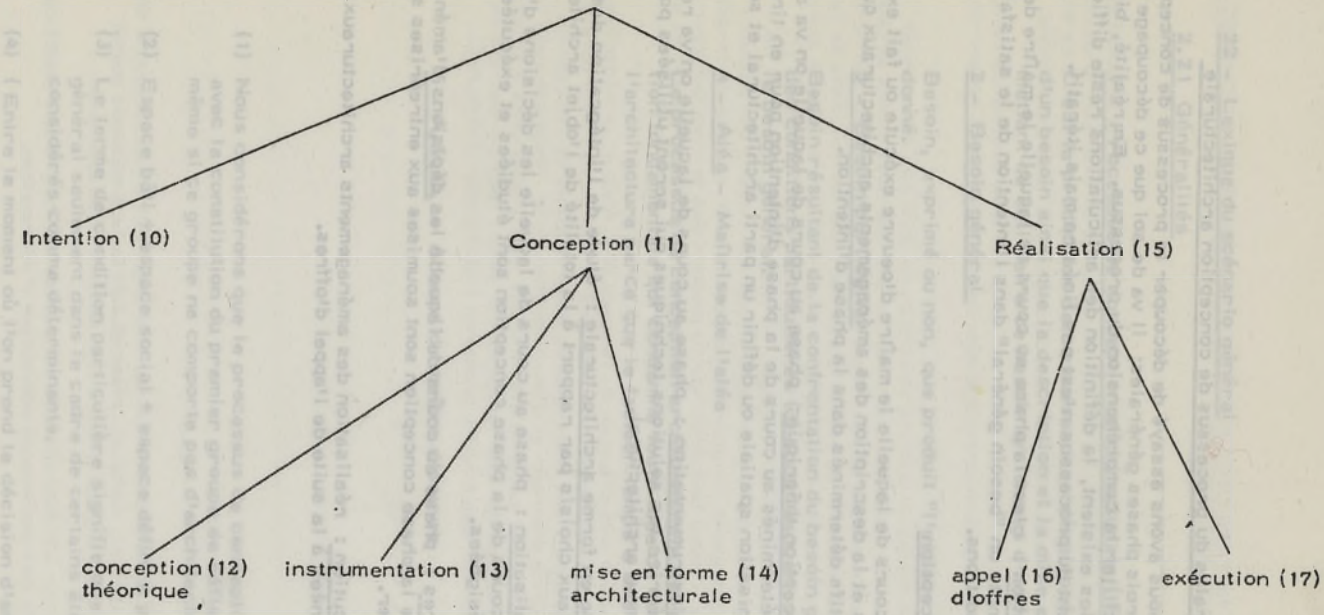
(14) Phase de mise en forme architecturale : étude de l'intégration des techniques et des matériaux choisis par rapport à la totalité de l'objet architectural.

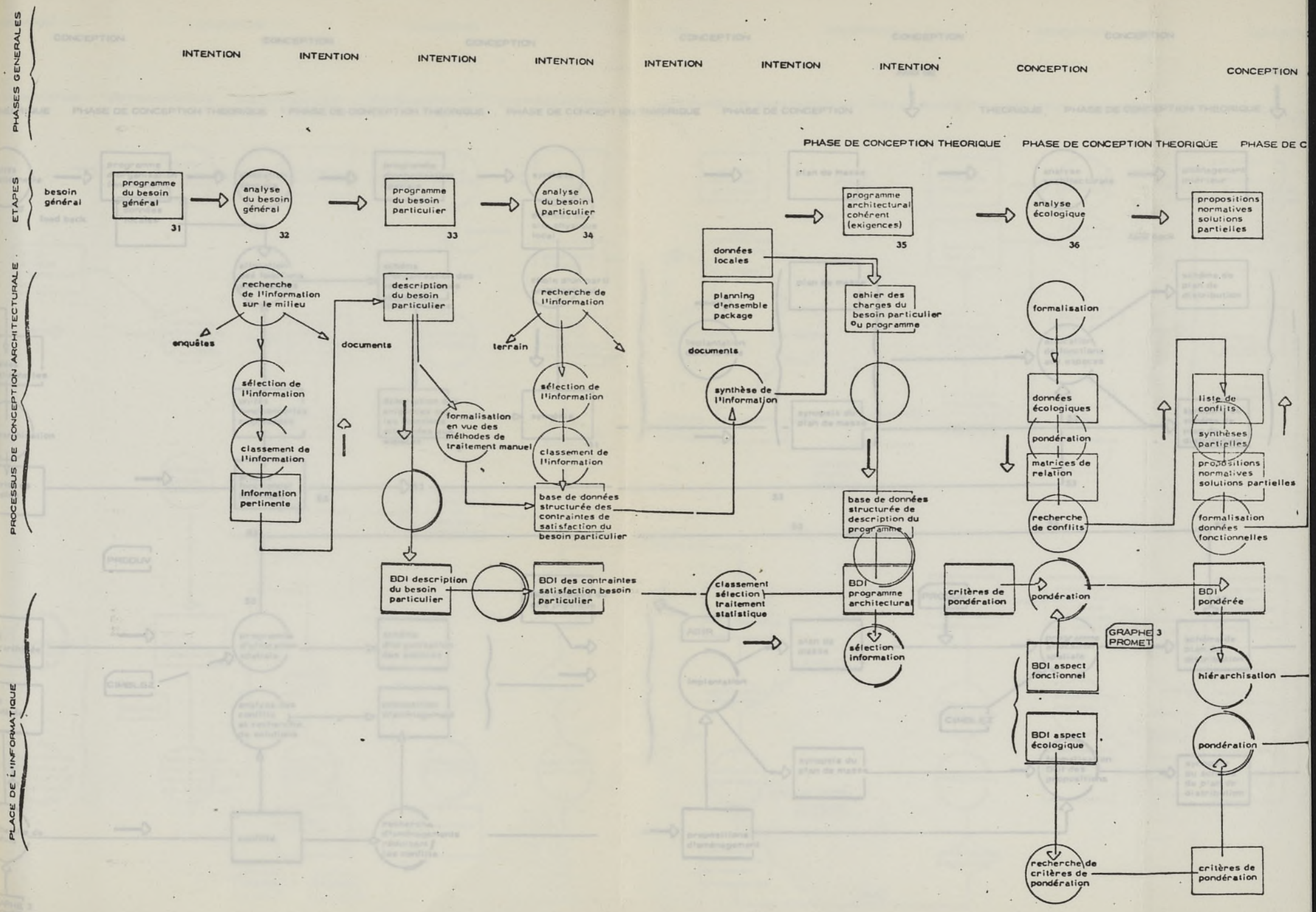
(15) Phase de réalisation : phase au cours de laquelle les décisions d'aménagement prises au cours de la phase conception sont étudiées et exécutées par les entreprises désignées.

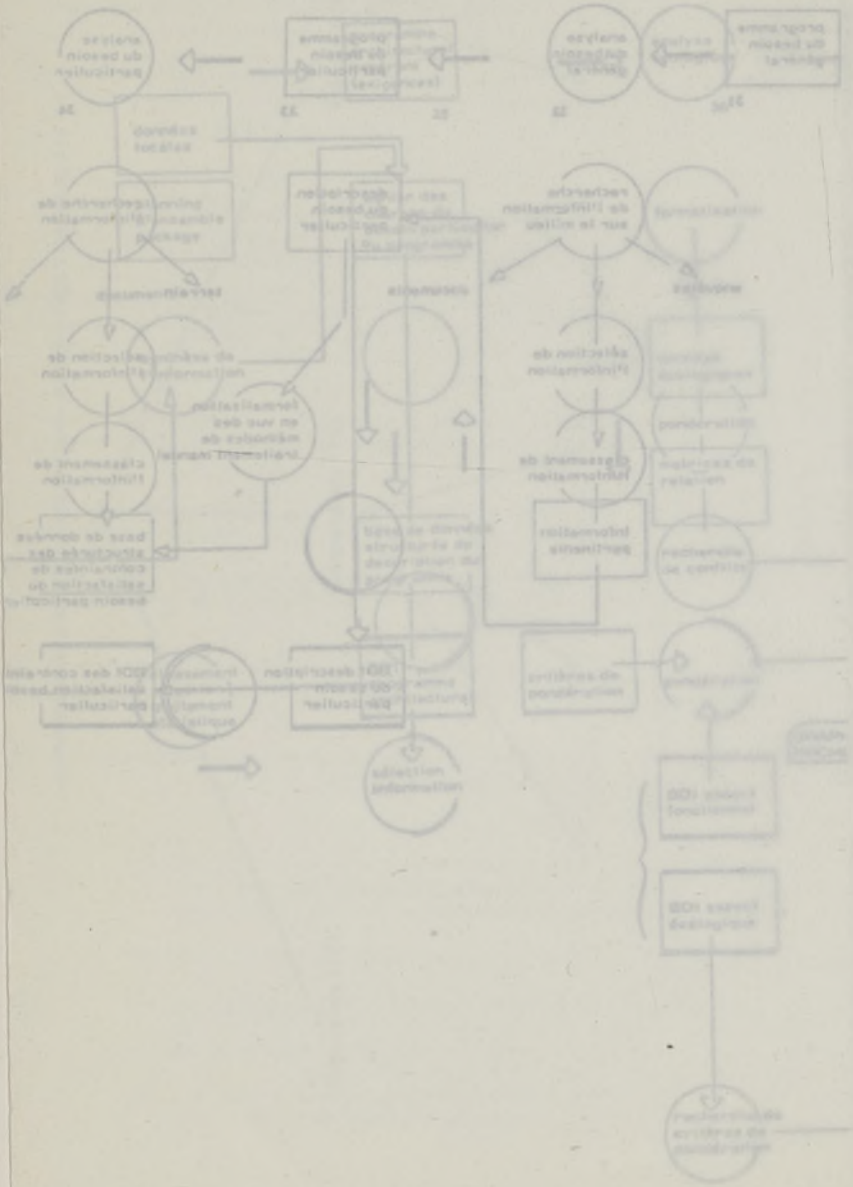
(16) Appels d'offres : phase au cours de laquelle les décisions d'aménagemen prises au cours de la phase conception sont soumises aux entreprises susceptibles de les réaliser.

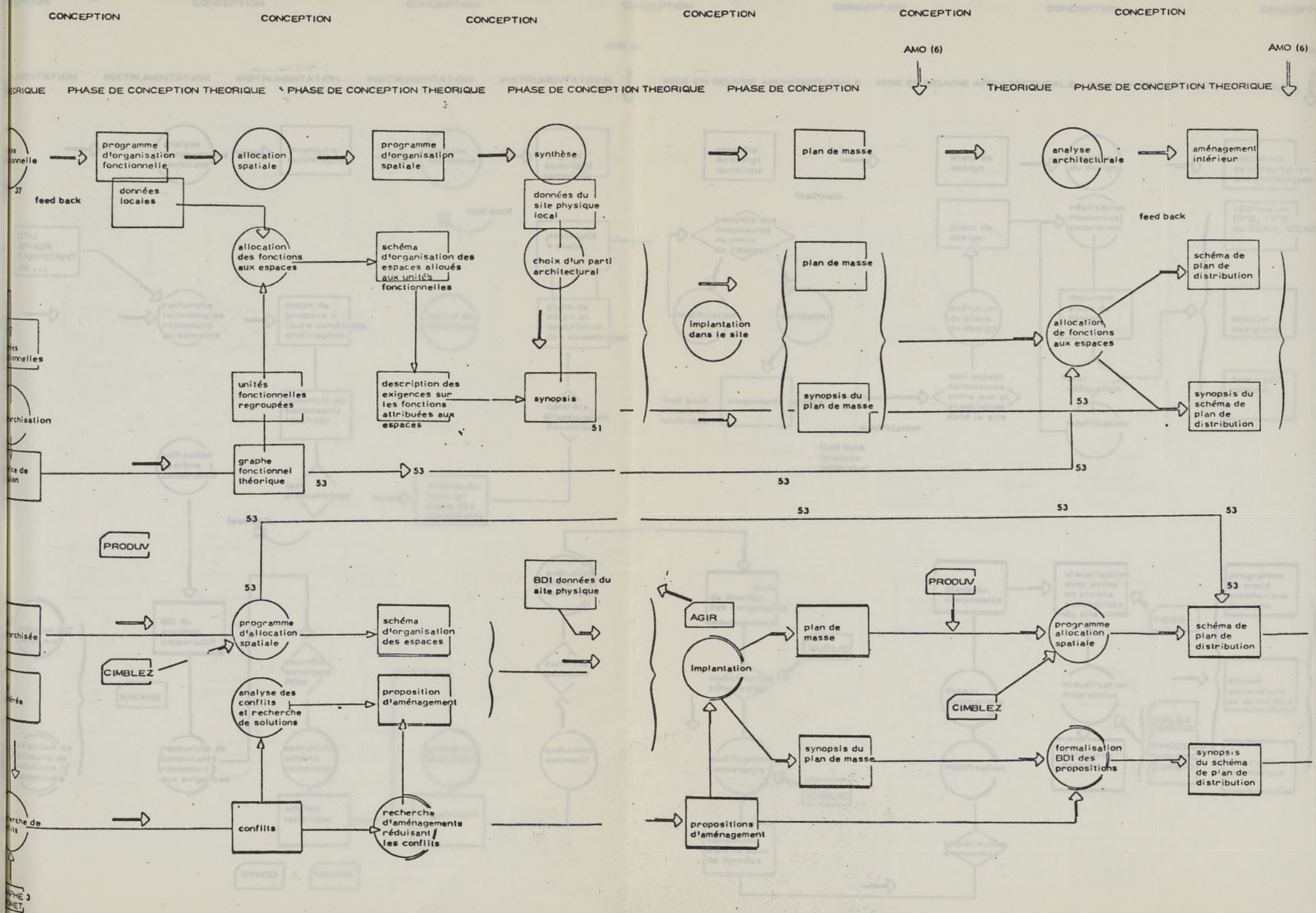
(17) Phase d'exécution : réalisation des aménagements architecturaux par les entreprises désignées à la suite de l'appel d'offres.

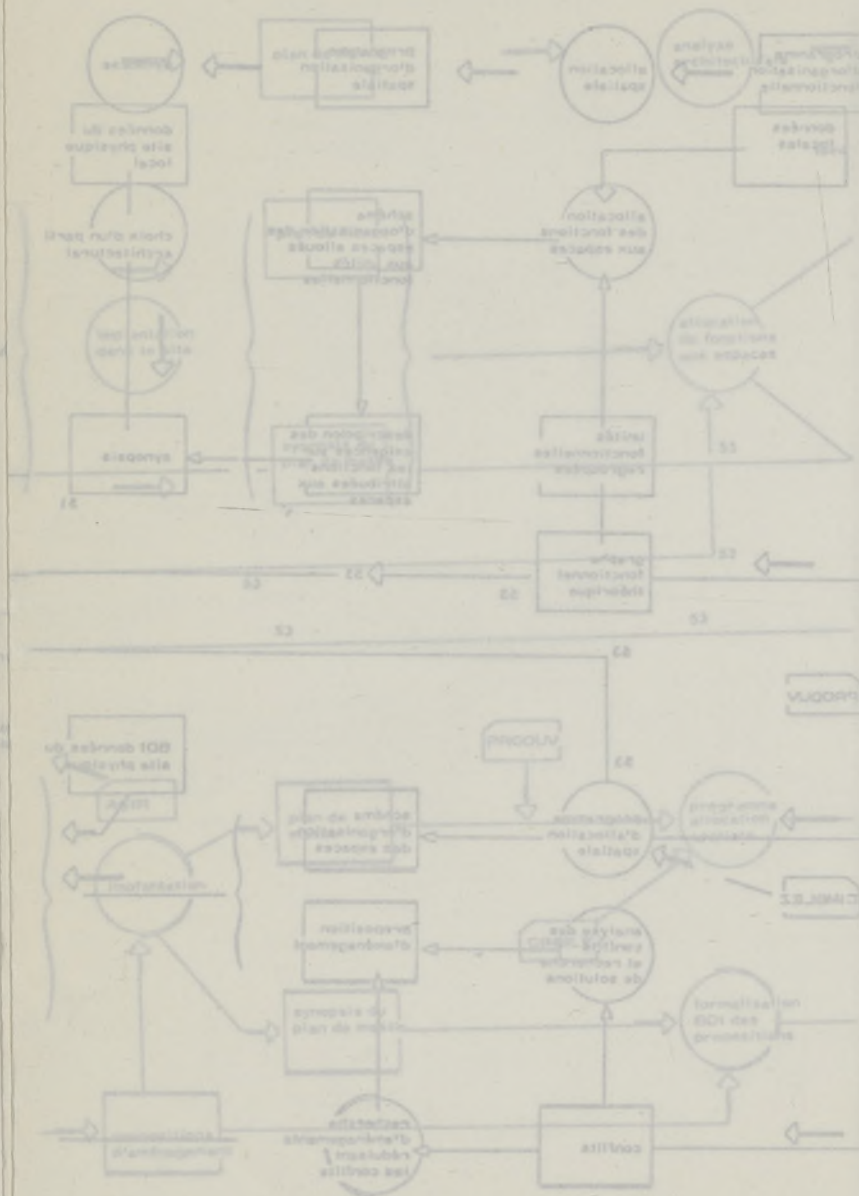
PROCESSUS DE CONCEPTION ARCHITECTURALE (1)

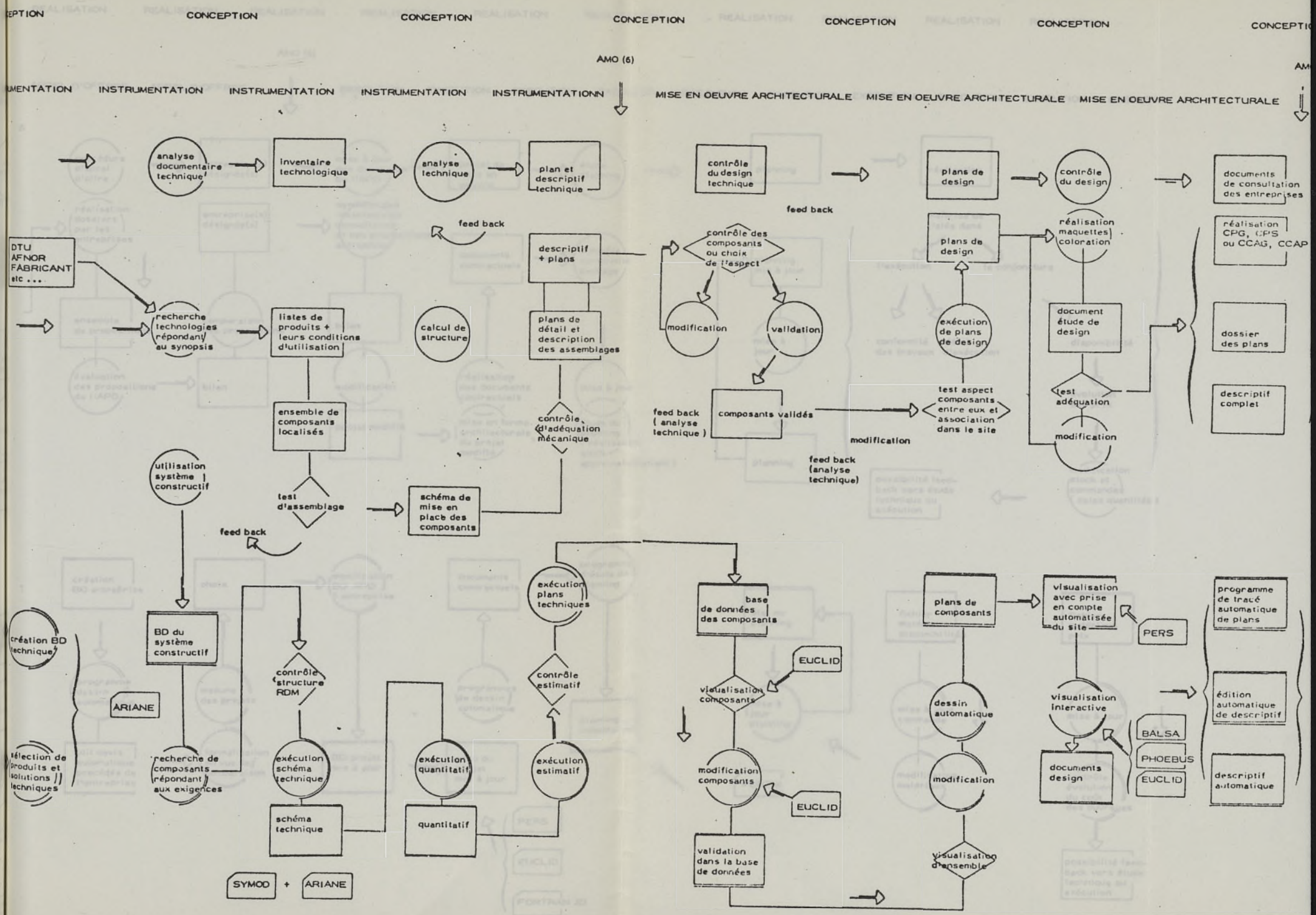


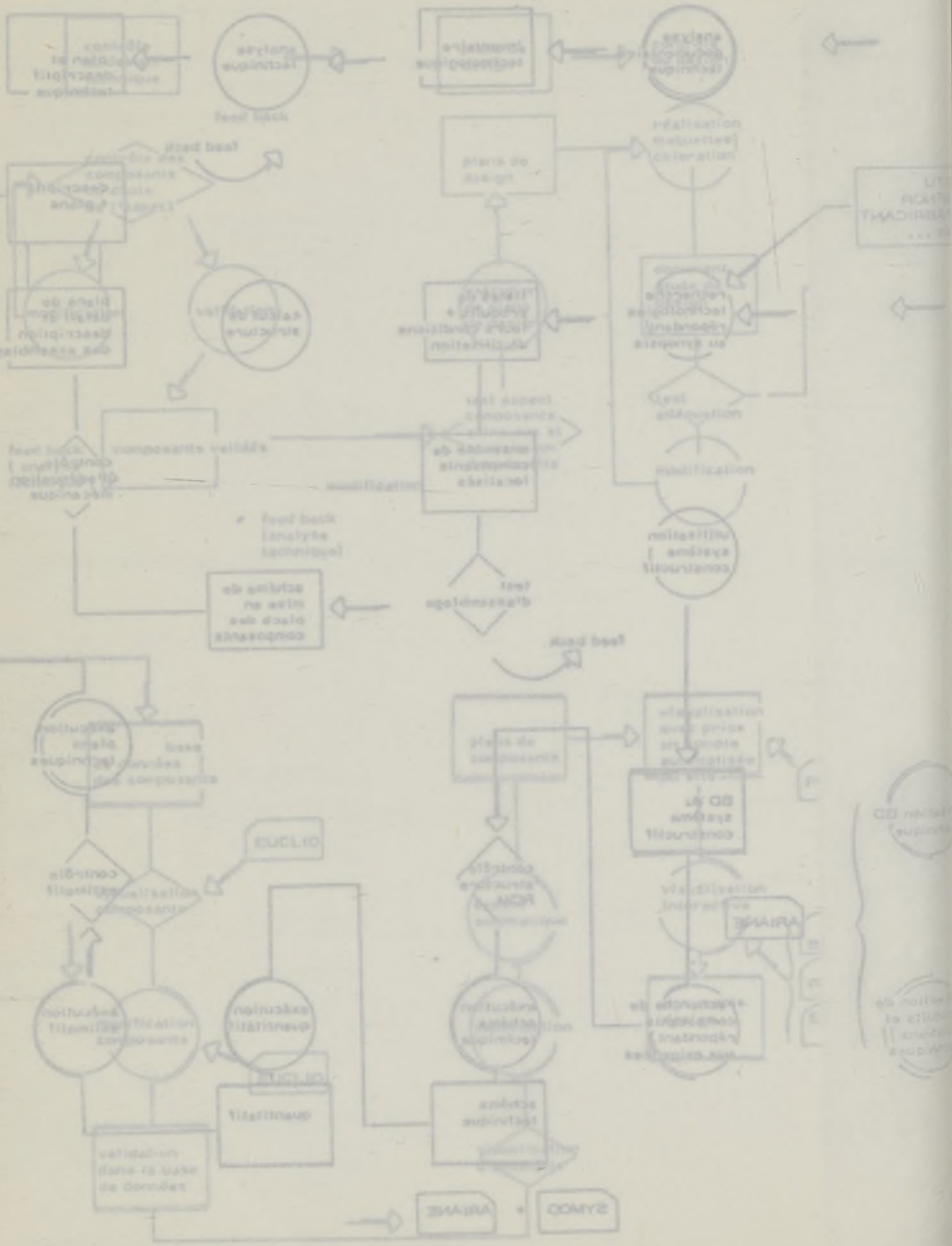

PHASES GENERALES
ETAPES
PROCESSUS DE CONCEPTION ARCHITECTURALE
PLACE DE L'INFORMATIQUE
INTENTION
INTENTION
INTENTION
INTENTION
INTENTION
INTENTION
INTENTION
CONCEPTION
CONCEPTION
PHASE DE CONCEPTION THEORIQUE
PHASE DE CONCEPTION THEORIQUE
besoin général
programme du besoin général
31
analyse du besoin général
32
programme du besoin particulier
33
analyse du besoin particulier
34
programme architectural cohérent (exigences)
35
analyse écologique
36
propositions normatives solutions partielles
recherche de l'information sur le milieu
enquêtes
documents
sélection de l'information
classement de l'information
information pertinente
description du besoin particulier
formalisation en vue des méthodes de traitement manuel
recherche de l'information
terrain
sélection de l'information
classement de l'information
base de données structurée des contraintes de satisfaction du besoin particulier
BDI description du besoin particulier
BDI des contraintes satisfaction besoin particulier
données locales
planning d'ensemble package
documents
synthèse de l'information
cahier des charges du besoin particulier ou programme
base de données structurée de description du programme
BDI programme architectural
classement sélection traitement statistique
sélection information
formalisation
données écologiques
pondération
matrices de relation
recherche de conflits
critères de pondération
pondération
BDI aspect fonctionnel
BDI aspect écologique
recherche de critères de pondération
GRAPHE PROMET 3
liste de conflits
synthèses partielles
propositions normatives solutions partielles
formalisation données fonctionnelles
BDI pondérée
hiérarchisation
pondération
critères de pondération

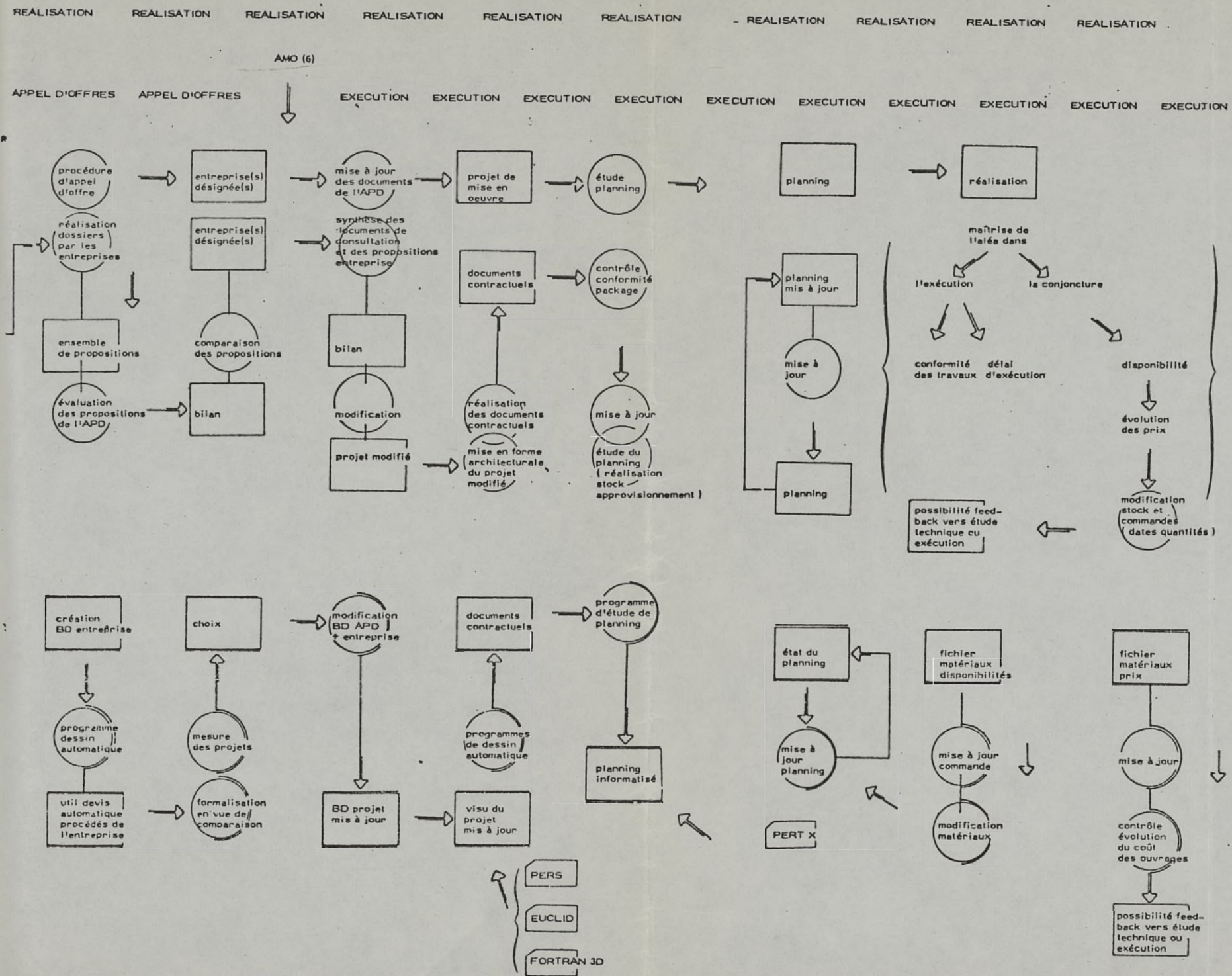


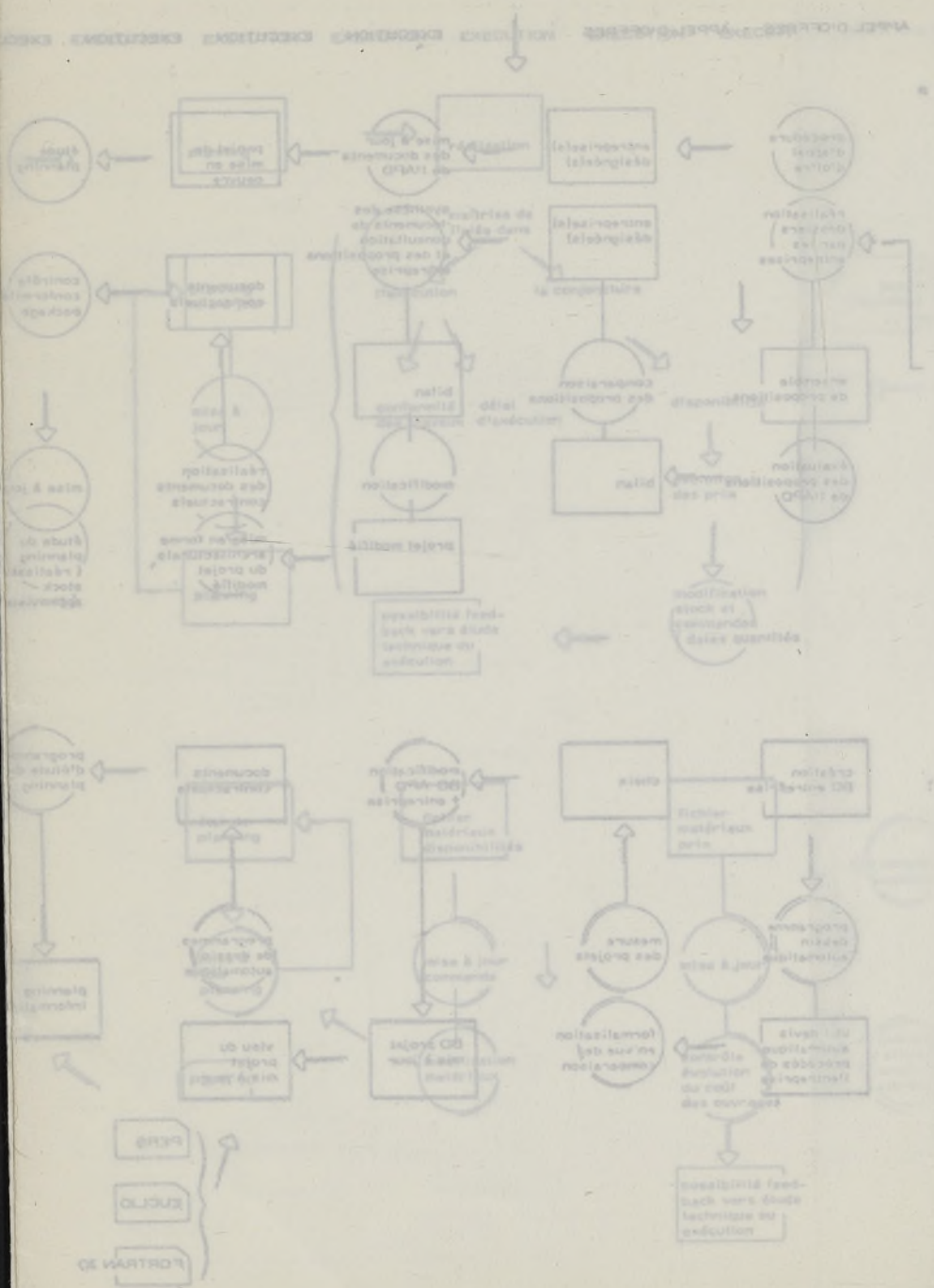

CONCEPTION
CONCEPTION
CONCEPTION
CONCEPTION
CONCEPTION
CONCEPTION
AMO (6)
AMO (6)
PHASE DE CONCEPTION THEORIQUE
PHASE DE CONCEPTION THEORIQUE
PHASE DE CONCEPTION THEORIQUE
PHASE DE CONCEPTION THEORIQUE
PHASE DE CONCEPTION
THEORIQUE
PHASE DE CONCEPTION THEORIQUE
programme d'organisation fonctionnelle
allocation spatiale
programme d'organisation spatiale
synthèse
plan de masse
analyse architecturale
aménagement intérieur
feed back
données locales
données du site physique local
feed back
allocation des fonctions aux espaces
schéma d'organisation des espaces alloués aux unités fonctionnelles
choix d'un parti architectural
plan de masse
schéma de plan de distribution
implantation dans le site
allocation de fonctions aux espaces
unités fonctionnelles regroupées
description des exigences sur les fonctions attribuées aux espaces
synopsis
synopsis du plan de masse
synopsis du schéma de plan de distribution
53
51
graphe fonctionnel théorique
53
53
53
53
53
53
53
PRODUV
53
BDI données du site physique
PRODUV
53
programme d'allocation spatiale
schéma d'organisation des espaces
AGIR
plan de masse
programme allocation spatiale
schéma de plan de distribution
CIMBLEZ
implantation
CIMBLEZ
analyse des conflits et recherche de solutions
proposition d'aménagement
synopsis du plan de masse
formalisation BDI des propositions
synopsis du schéma de plan de distribution
conflits
recherche d'aménagements réduisant les conflits
propositions d'aménagement




CONCEPTION
CONCEPTION
CONCEPTION
CONCEPTION
CONCEPTION
CONCEPTION
INSTRUMENTATION
INSTRUMENTATION
INSTRUMENTATION
INSTRUMENTATION
INSTRUMENTATIONN
AMO (6)
MISE EN OEUVRE ARCHITECTURALE
MISE EN OEUVRE ARCHITECTURALE
MISE EN OEUVRE ARCHITECTURALE
analyse documentaire technique
inventaire technologique
analyse technique
plan et descriptif technique
contrôle du design technique
plans de design
contrôle du design
documents de consultation des entreprises
feed back
DTU AFNOR FABRICANT etc ...
descriptif + plans
contrôle des composants ou choix de l'aspect
plans de design
réalisation maquettes coloration
réalisation CPG, CPS ou CCAG, CCAP
recherche technologies répondant au synopsis
listes de produits + leurs conditions d'utilisation
calcul de structure
plans de détail et description des assemblages
modification
validation
exécution de plans de design
document étude de design
dossier des plans
ensemble de composants localisés
contrôle d'adéquation mécanique
feed back (analyse technique)
composants validés
test aspect composants entre eux et association dans le site
test adéquation
descriptif complet
utilisation système constructif
test d'assemblage
schéma de mise en place des composants
modification
feed back
feed back (analyse technique)
exécution plans techniques
base de données des composants
plans de composants
visualisation avec prise en compte automatisée du site
programme de tracé automatique de plans
création BD technique
BD du système constructif
contrôle structure RDM
contrôle estimatif
EUCLID
PERS
ARIANE
visualisation composants
dessin automatique
visualisation interactive
édition automatique de descriptif
sélection de produits et solutions techniques
recherche de composants répondant aux exigences
exécution schéma technique
exécution quantitatif
exécution estimatif
modification composants
modification
documents design
BALSA
PHOEBUS
EUCLID
descriptif automatique
schéma technique
quantitatif
validation dans la base de données
visualisation d'ensemble
SYMOD + ARIANE




REALISATION
REALISATION
REALISATION
REALISATION
REALISATION
REALISATION
REALISATION
REALISATION
REALISATION
REALISATION
AMO (6)
APPEL D'OFFRES
APPEL D'OFFRES
EXECUTION
EXECUTION
EXECUTION
EXECUTION
EXECUTION
EXECUTION
EXECUTION
EXECUTION
EXECUTION
EXECUTION
procédure d'appel d'offre
entreprise(s) désignée(s)
mise à jour des documents de l'APD
projet de mise en oeuvre
étude planning
planning
réalisation
réalisation dossiers par les entreprises
entreprise(s) désignée(s)
synthèse des documents de consultation et des propositions entreprise
documents contractuels
contrôle conformité package
planning mis à jour
maîtrise de l'aléa dans
l'exécution
la conjoncture
ensemble de propositions
comparaison des propositions
bilan
modification
réalisation des documents contractuels
mise à jour
mise à jour
conformité des travaux
délai d'exécution
disponibilité
évaluation des propositions de l'APD
bilan
projet modifié
mise en forme architecturale du projet modifié
étude du planning (réalisation stock approvisionnement)
planning
évolution des prix
possibilité feed-back vers étude technique ou exécution
modification stock et commandes (dates quantités)
création BD entreprise
choix
modification BD APD + entreprise
documents contractuels
programme d'étude de planning
état du planning
fichier matériaux disponibilités
fichier matériaux prix
programme dessin automatique
mesure des projets
programmes de dessin automatique
mise à jour planning
mise à jour commande
mise à jour
util devis automatique procédés de l'entreprise
formalisation en vue de comparaison
BD projet mis à jour
visu du projet mis à jour
planning informatisé
PERT X
modification matériaux
contrôle évolution du coût des ouvrages
possibilité feed-back vers étude technique ou exécution
PERS
EUCLID
FORTRAN 3D



2.23 - Etapes du processus

(31) Programme du besoin général : document émanant du maître d'oeuvre ou de groupes de travail dans lequel le besoin général est décrit en terme d'objectifs généraux à atteindre.

(32) Analyse du besoin général : recherche et prise en compte des informations particulières signifiantes quant à la satisfaction du besoin général en terme d'aménagement, classement et synthèse de ces données en vue d'aboutir à l'expression d'un besoin particulier.

(33) Programme d'un besoin particulier : document décrivant le besoin particulier défini à partir des conditions particulières de satisfaction du besoin général.

(34) Analyse du besoin particulier : prise en compte des données déterminantes locales (conditions de réalisation, juridiques, économiques, techniques, etc ...).

(35) Programme cohérent : ensemble d'exigences (1) en terme d'aménagement que nécessite la satisfaction du besoin particulier sous la contrainte des données déterminantes (voir exemple Annexe 2)

(36) Analyse écologique(2) formalisation des relations entre espaces après études des rapports qualitatifs entre ces espaces.

(37) Analyse fonctionnelle (2) : recherche des informations pertinentes déterminantes pour l'allocation spaciale.

(1) Les exigences sont déterminées à partir d'une synthèse partielle des données déterminantes.

(2) Lorsque l'analyse fonctionnelle ne nécessite pas une analyse écologique préalable c'est que celle-ci est implicite, c'est-à-dire que les informations concernées sont déjà formalisées dans un contexte indépendant du P.C.A. concerné.

2.24 - Description du processus

(51) Synopsis : document écrit qui complète les documents graphiques; ce document définit le quoi des objectifs à réaliser sans préjuger du comment. Il y a un synopsis pour chaque évènement du processus de conception.

(52) Synopsis du programme architectural cohérent : définition des aménagements que l'on doit faire pour satisfaire le programme architectural cohérent c'est-à-dire la définition précise mais en termes théoriques des objectifs architecturaux à réaliser.

(53) Nous avons voulu montrer que suivant le type d'étude l'étape allocation spatiale pouvait soit être réalisée en une étape conduisant directement au schéma de plan de distribution, soit en deux étapes (plan de masse, schéma de plan de distribution).

3 - UTILISATION D'ARLANG DANS LE CADRE METHODOLOGIQUE PROPOSE :

Nous reprenons dans ce chapitre le processus méthodologique de conception proposé en chapitre 2, afin de montrer la place et l'intérêt de l'utilisation du langage ARLANG pour l'élaboration d'un projet.

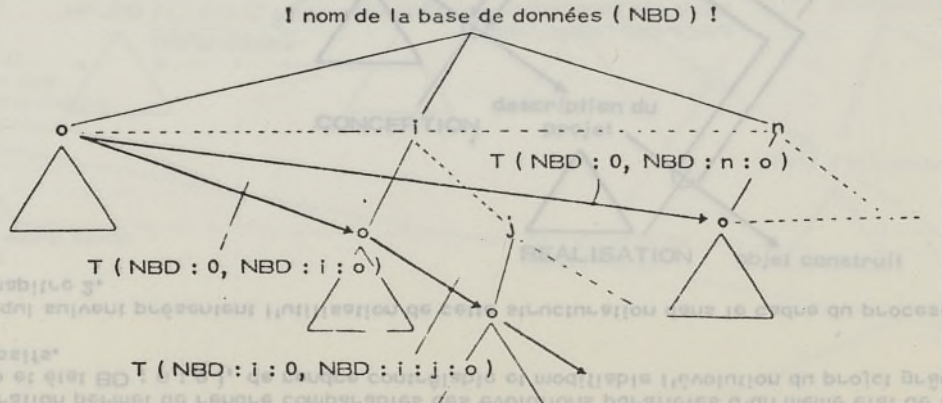
L'utilisation du langage sera examinée selon trois volets :

- Bases de données et états de base de données
- Programme de traitement
- Description d'informations au sein d'un état de base de données

Chacun de ces points sera exposé sous forme de schémas théoriques suivis d'exemples concrets d'illustration.

31 - Bases de données et états de base de données :

Nous avons, dans le rapport définitif du contrat d'élaboration du langage ARLANG, exposé la notion de "base de données" et d'"état" de base de données (ch. 1, § 22 - ch. 2, 1121). Rappelons qu'une base de données est constituée d'états; ces états représentant les situations successives ou parallèles d'une base de données au cours des modifications qui lui sont appliquées, le schéma théorique de structuration des divers états d'une base de données est le suivant :



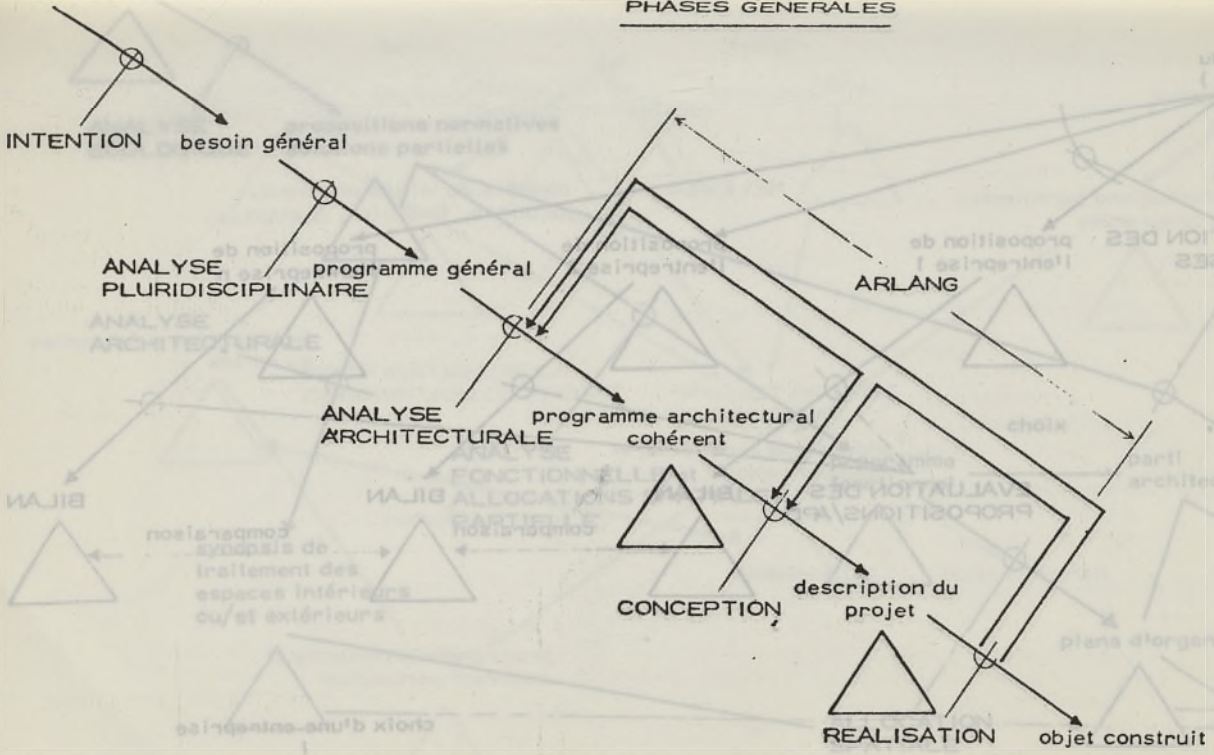
T (N, M) : ensemble des programmes ARLANG qui ont permis de passer de l'état N à l'état M

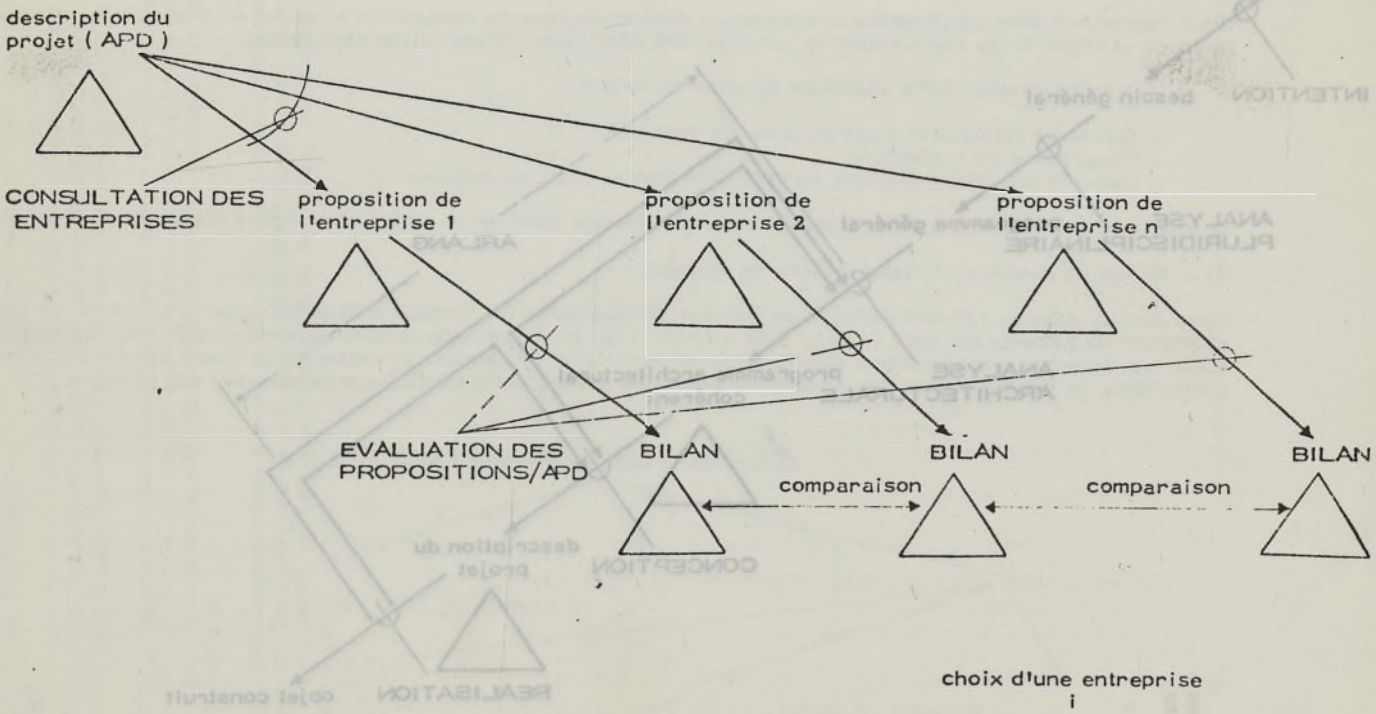
Cette structuration permet de rendre comparables des évolutions parallèles d'un même état de la base (exemple : état BD : i : o et état BD : n : o), de rendre contrôlable et modifiable l'évolution du projet grâce à la conservation d'états successifs.

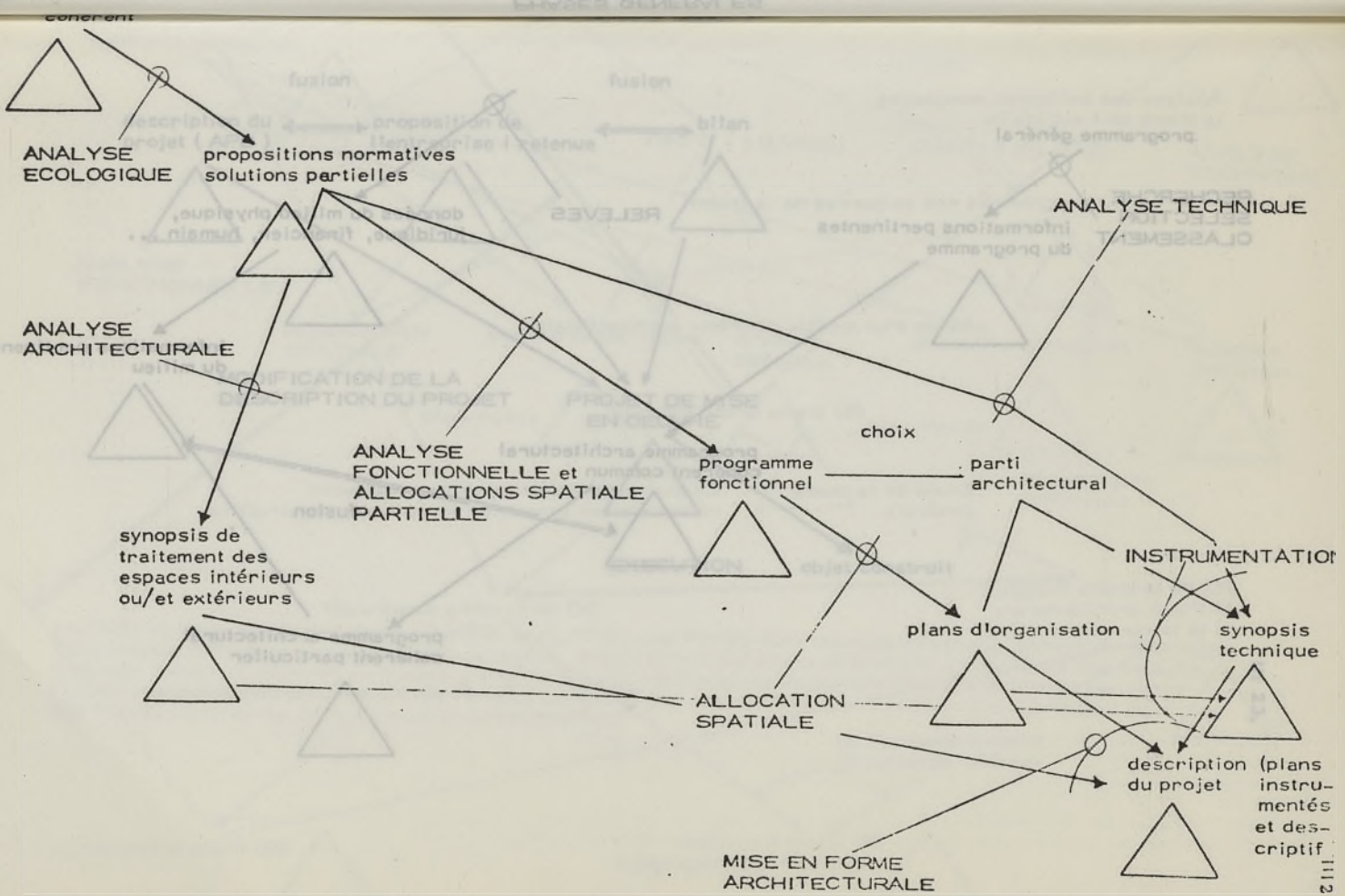
Les schémas qui suivent présentent l'utilisation de cette structuration dans le cadre du processus méthodologique proposé au chapitre 2.

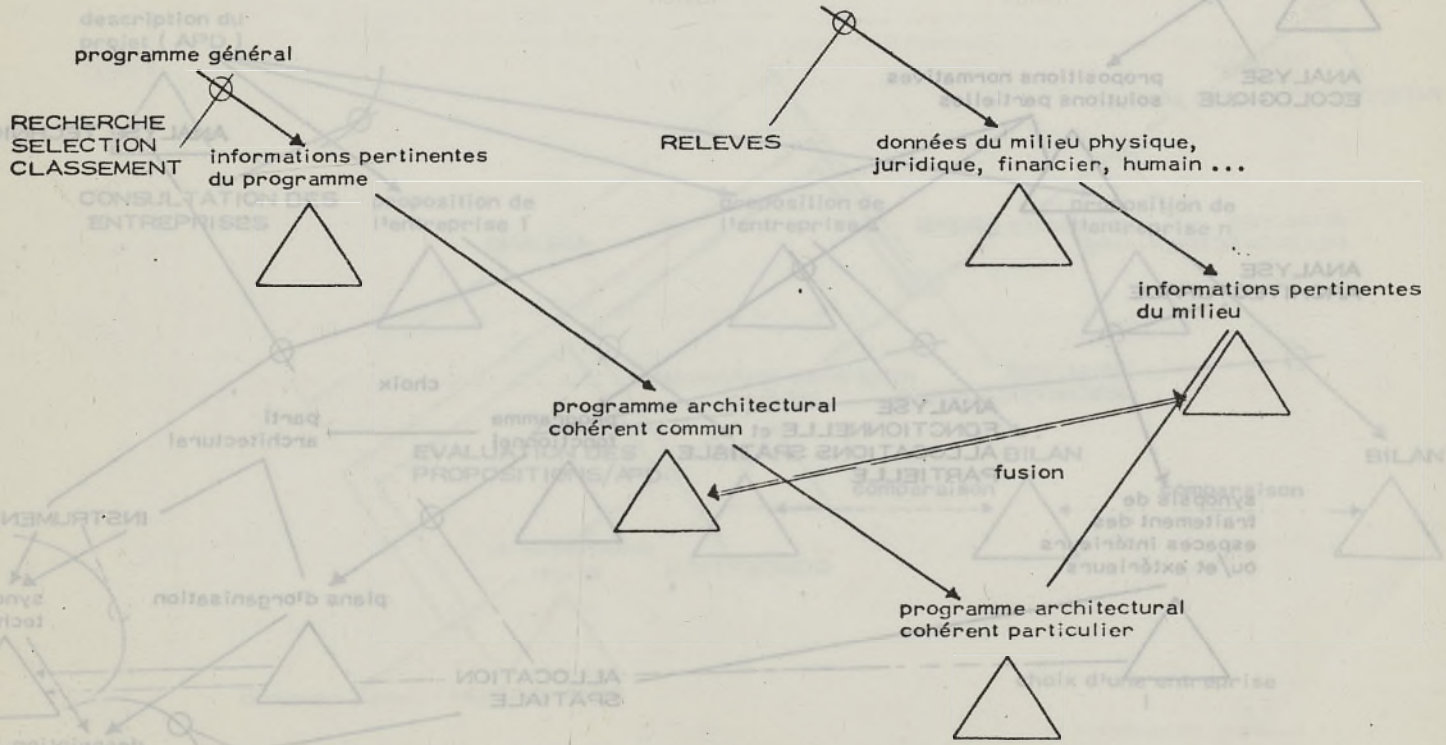
PHASES GENERALES

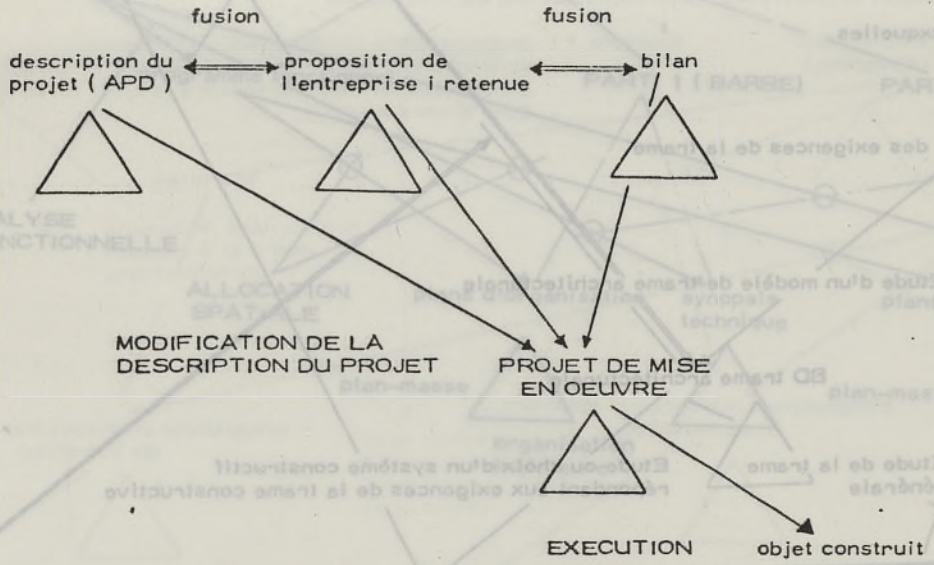
PHASES GENERALES

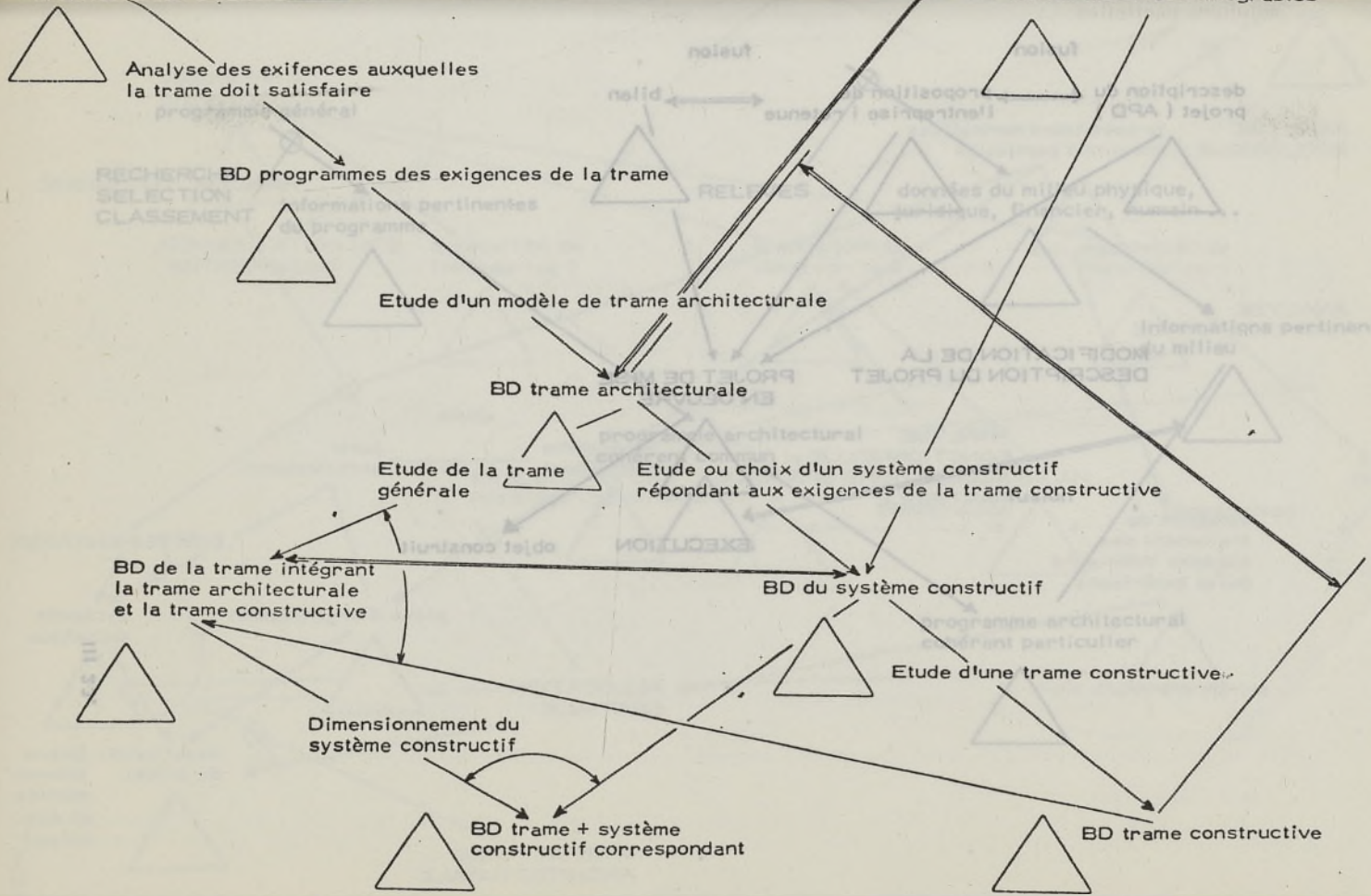



description du
projet (APD)
CONSULTATION DES
ENTREPRISES
proposition de
l'entreprise 1
proposition de
l'entreprise 2
proposition de
l'entreprise n
EVALUATION DES
PROPOSITIONS/APD
BILAN
BILAN
BILAN
comparaison
comparaison
choix d'une entreprise
i

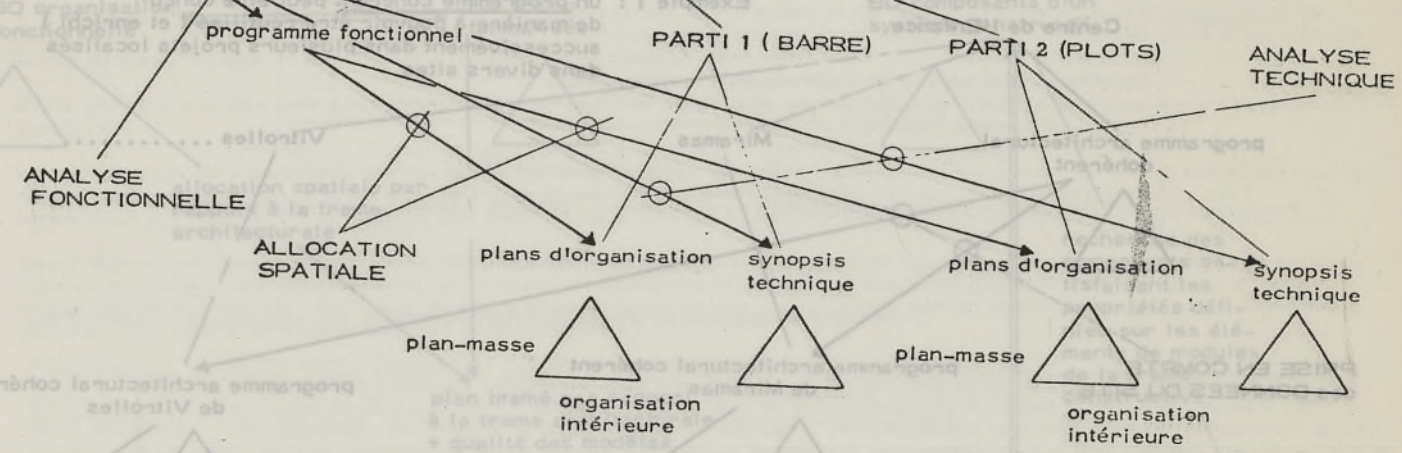

cohérent
ANALYSE ECOLOGIQUE
propositions normatives
solutions partielles
ANALYSE TECHNIQUE
ANALYSE ARCHITECTURALE
ANALYSE FONCTIONNELLE et ALLOCATIONS SPATIALE PARTIELLE
choix
programme fonctionnel
parti architectural
synopsis de traitement des espaces intérieurs ou/et extérieurs
INSTRUMENTATION
plans d'organisation
synopsis technique
ALLOCATION SPATIALE
description du projet
(plans instrumentés et descriptif
MISE EN FORME ARCHITECTURALE


programme général
RECHERCHE
SELECTION
CLASSEMENT
informations pertinentes
du programme
RELEVES
données du milieu physique,
juridique, financier, humain ...
informations pertinentes
du milieu
programme architectural
cohérent commun
fusion
programme architectural
cohérent particulier


fusion
fusion
description du projet (APD)
proposition de l'entreprise i retenue
bilan
MODIFICATION DE LA DESCRIPTION DU PROJET
PROJET DE MISE EN OEUVRE
EXECUTION
objet construit


Analyse des exifences auxquelles
la trame doit satisfaire
BD programmes des exigences de la trame
Etude d'un modèle de trame architecturale
BD trame architecturale
Etude de la trame
générale
Etude ou choix d'un système constructif
répondant aux exigences de la trame constructive
BD de la trame intégrant
la trame architecturale
et la trame constructive
BD du système constructif
Etude d'une trame constructive
Dimensionnement du
système constructif
BD trame + système
constructif correspondant
BD trame constructive

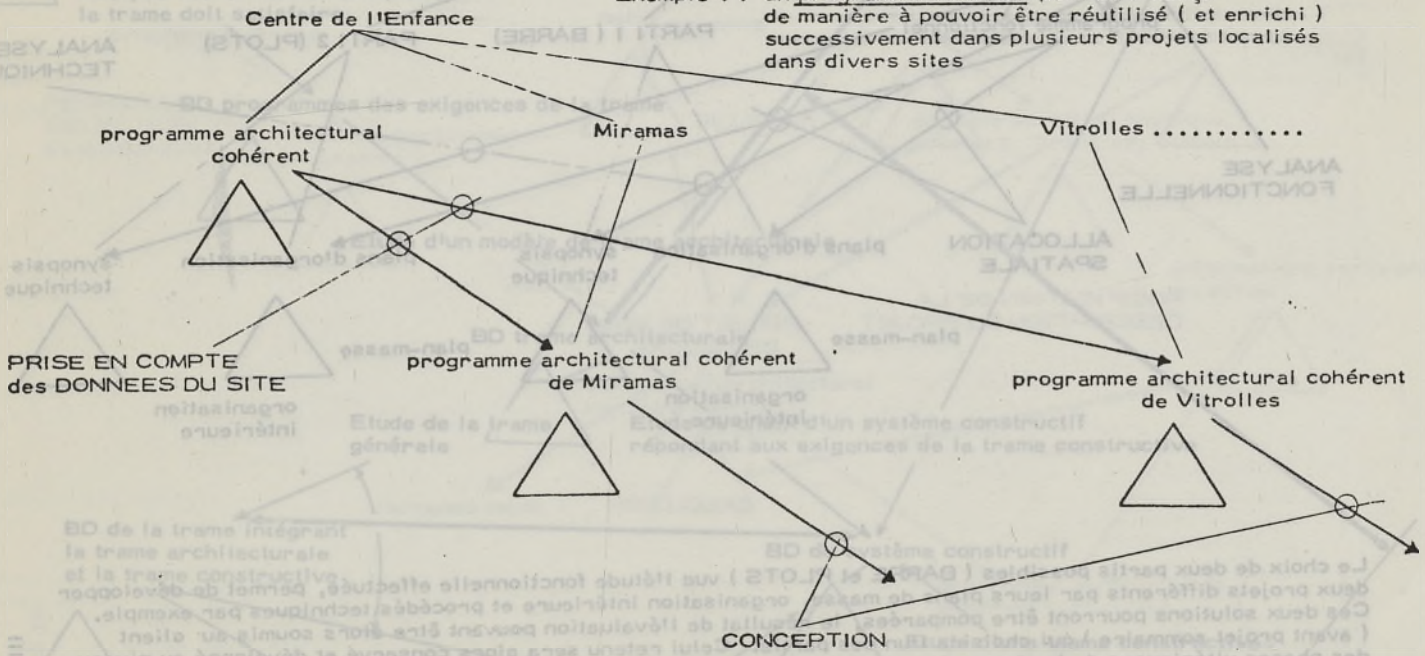
propositions normatives
solutions partielles



Le choix de deux partis possibles (BARRE et PLOTS) vue l'étude fonctionnelle effectuée, permet de développer deux projets différents par leurs plans de masse, organisation intérieure et procédés techniques par exemple. Ces deux solutions pourront être comparées, le résultat de l'évaluation pouvant être alors soumis au client (avant projet sommaire) qui choisira l'un des partis. Celui retenu sera alors conservé et développé au niveau des phases ultérieures de la conception et réalisation.

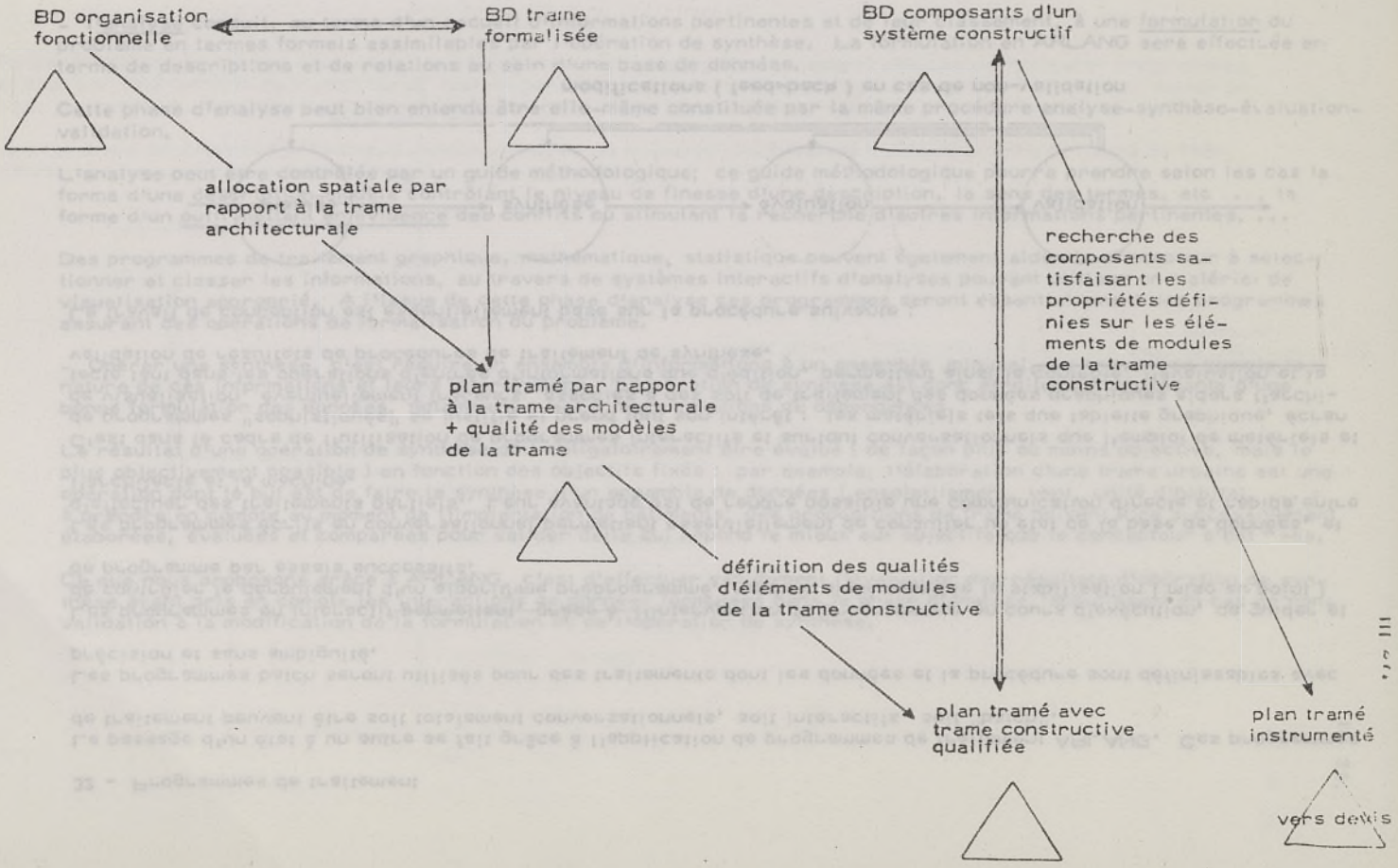
REUTILISATION D'UNE MEME BASE DE DONNEES

Exemple 1 : un programme cohérent peut être conçu de manière à pouvoir être réutilisé (et enrichi) successivement dans plusieurs projets localisés dans divers sites



- Le programme architectural cohérent du site est un enrichissement du programme architectural cohérent commun.
  Ce programme architectural commun ne traite que des données communes aux divers sites (données humaines, régionales ...
  Le programme particulier à un site le complète par les données relatives aux conditions particulières du milieu

- On peut ainsi en ARLANG bénéficier d'un travail déjà effectué dans une opération similaire (Centre de l'Enfance) et enrichir au fur et à mesure de son utilisation la base de données communes


BD organisation fonctionnelle
BD trame formalisée
BD composants d'un système constructif
allocation spatiale par rapport à la trame architecturale
recherche des composants satisfaisant les propriétés définies sur les éléments de modules de la trame constructive
plan tramé par rapport à la trame architecturale + qualité des modèles de la trame
définition des qualités d'éléments de modules de la trame constructive
plan tramé avec trame constructive qualifiée
plan tramé instrumenté
vers deux

32 - Programmes de traitement

Le passage d'un état à un autre se fait grâce à l'application de programmes de traitement ARLANG. Ces programmes de traitement peuvent être soit totalement conversationnels, soit interactifs, soit "batch".

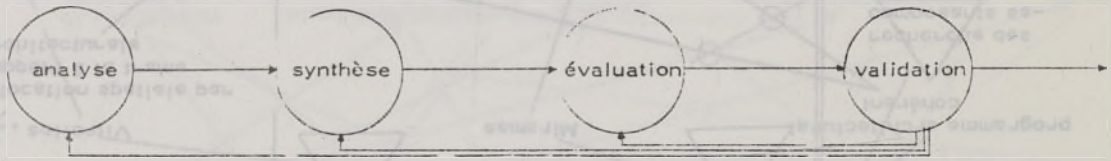
Les programmes batch seront utilisés pour des traitements dont les données et la procédure sont définissables avec précision et sans ambiguité.

Les programmes en interactif permettent, grâce à l'intervention du programmeur en cours d'exécution, de guider et de contrôler le déroulement d'un algorithme préprogrammé, rendant ainsi possible la stabilisation (mise au point) de programme par essais successifs.

Les programmes écrits en conversationnel permettent essentiellement de consulter un état de la base de données, et d'effectuer des traitements partiels. Leur avantage est de rendre possible une communication directe et rapide entre l'architecte et la machine.

C'est dans le cadre de l'utilisation de programmes interactifs et surtout conversationnels que l'emploi de matériels et de programmes "sophistiqués" se justifie et prend tout son intérêt : les matériels tels que tablette graphique, écran de visualisation, éventuellement traceurs, associés à des soft de traitement des données graphiques aidera l'architecte tant dans les opérations d'entrée d'informations que d'édition, permettant ainsi le contrôle, l'évaluation et la validation de résultats de procédures de traitement de synthèse.

Le travail de conception est essentiellement basé sur la procédure suivante :



- l'analyse conduit, au terme d'un recueil d'informations pertinentes et de leur classement, à une formulation du problème en termes formels assimilables par l'opération de synthèse. La formulation en ARLANG sera effectuée en terme de descriptions et de relations au sein d'une base de données.

Cette phase d'analyse peut bien entendu être elle-même constituée par la même procédure analyse-synthèse-évaluation-validation.

L'analyse peut être contrôlée par un guide méthodologique; ce guide méthodologique pourra prendre selon les cas la forme d'une description-modèle contrôlant le niveau de finesse d'une description, le sens des termes, etc ..., la forme d'un outil mettant en évidence des conflits ou stimulant la recherche d'autres informations pertinentes, ...

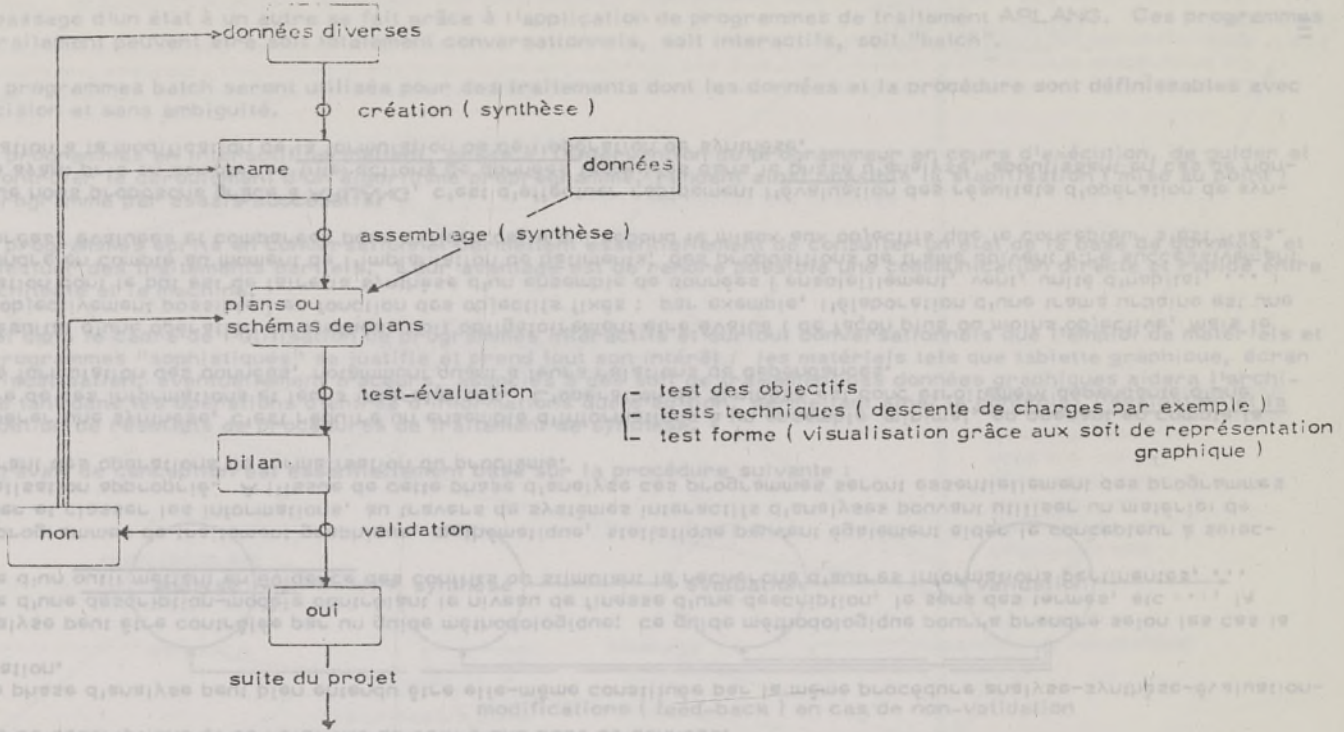
Des programmes de traitement graphique, mathématique, statistique peuvent également aider le concepteur à sélectionner et classer les informations, au travers de systèmes interactifs d'analyses pouvant utiliser un matériel de visualisation approprié. A l'issue de cette phase d'analyse ces programmes seront essentiellement des programmes assurant des opérations de formalisation du problème.

- Opérer une synthèse, c'est réduire un ensemble d'informations à un ensemble minimal, en prenant en compte la nature de ces informations et leurs interelations. L'opération de synthèse est donc étroitement dépendante d'une bonne formulation des données, notamment quant à leurs relations de dépendances.

Le résultat d'une opération de synthèse doit obligatoirement être évalué (de façon plus ou moins objective, mais le plus objectivement possible) en fonction des objectifs fixés : par exemple, l'élaboration d'une trame urbaine est une opération dont le but est de faire la synthèse d'un ensemble de données (ensoleillement, vent, unité d'habitat, ...) à prendre en compte au moment de l'implantation de bâtiments; des propositions de trame doivent être successivement élaborées, évaluées et comparées pour valider celle qui répond le mieux aux objectifs que le concepteur s'est fixés.

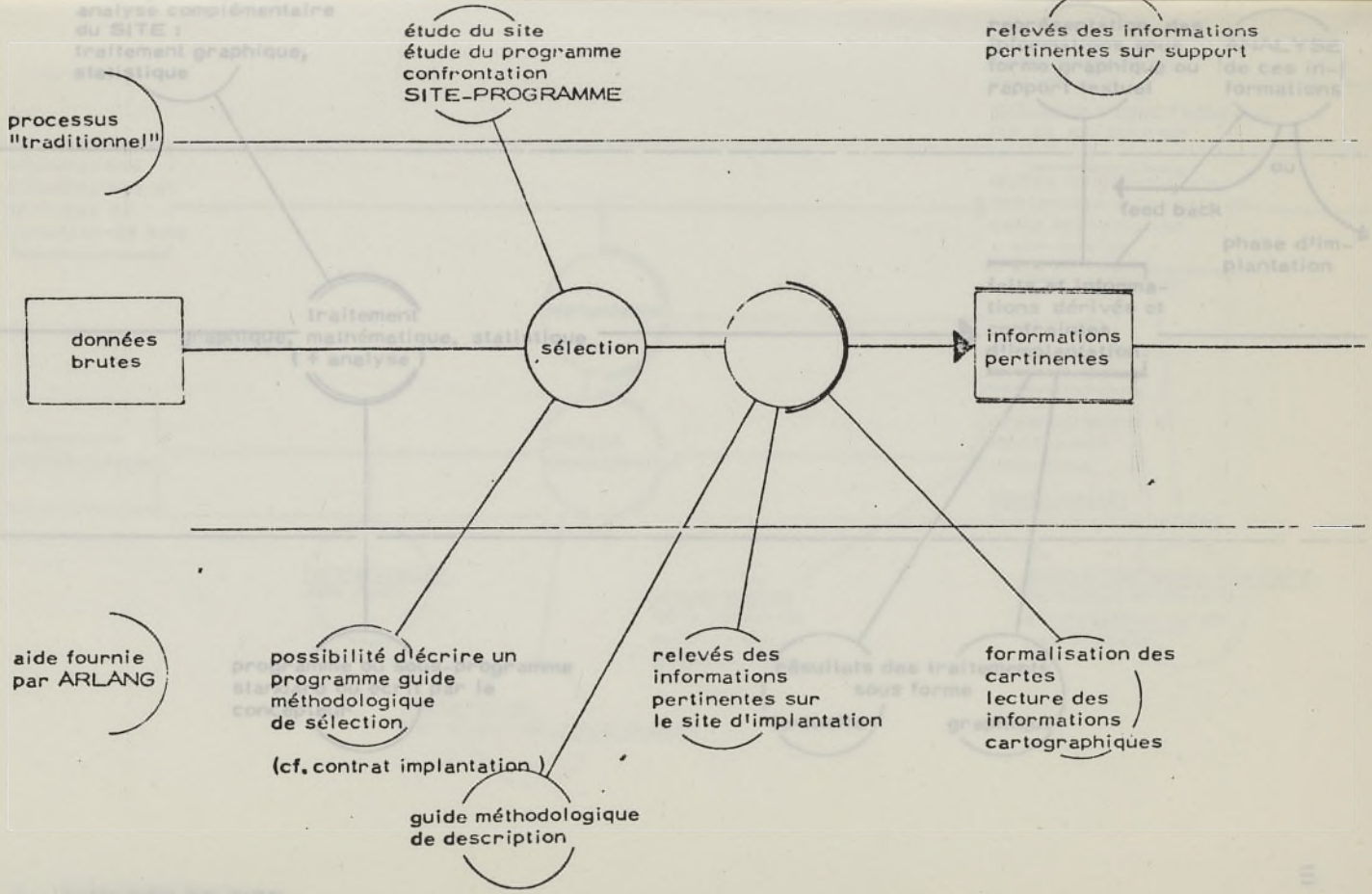
Ce que nous proposons grâce à ARLANG, c'est d'effectuer rapidement l'évaluation des résultats d'opération de synthèse ayant pris en compte les interactions de données formulées dans le phase d'analyse, aboutissant en cas de non-validation à la modification de la formulation ou de l'opération de synthèse.

Exemple de procédure :



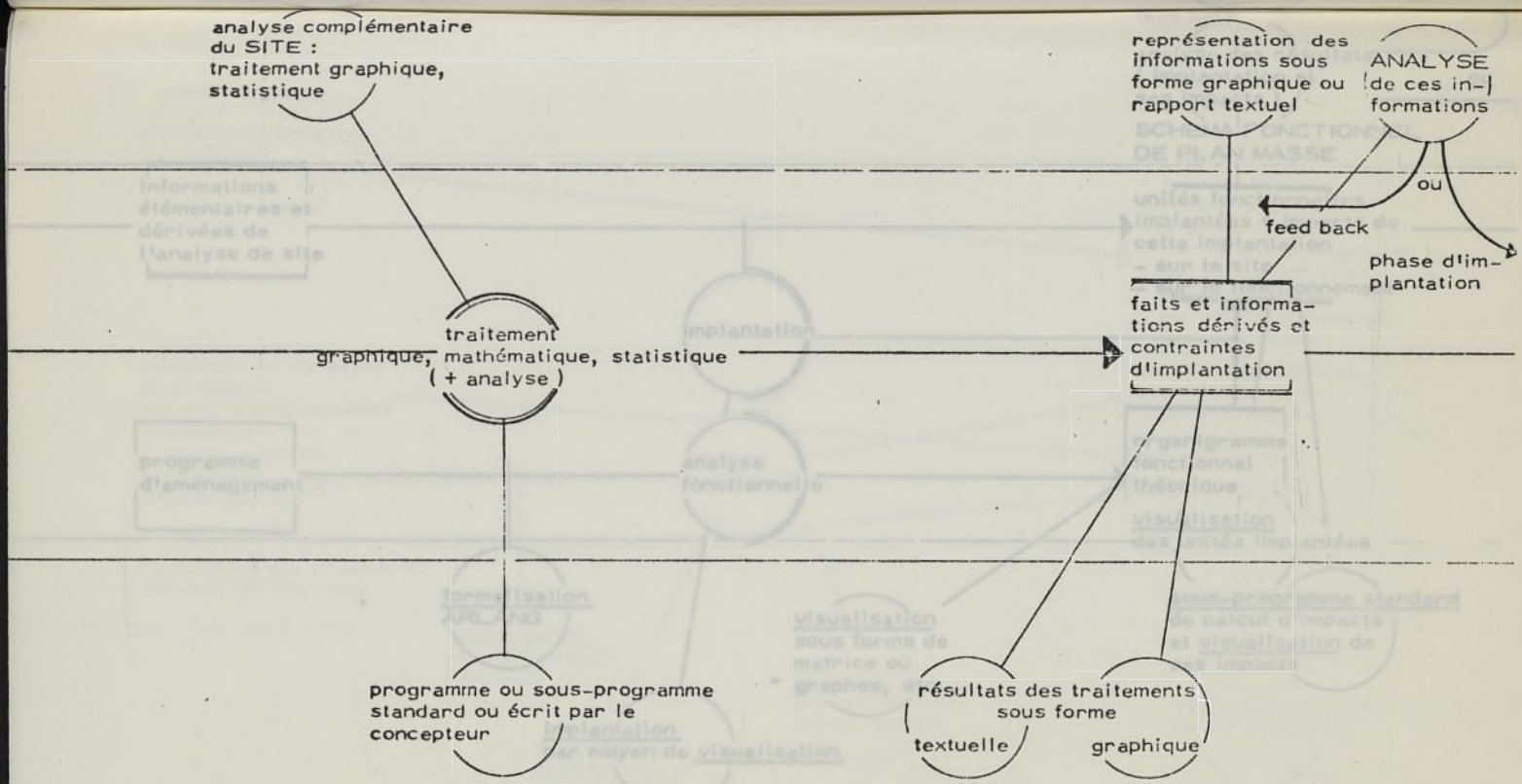
Comme le montre le schéma ci-dessus, le contrôle peut être fait grâce à l'emploi d'outils de "calculs" programmés, ou bien par intervention de l'architecte, testant par exemple la pertinence d'une forme grâce à une visualisation en plan, perspective, donc dans le cadre d'un jugement basé sur des critères "subjectifs" (non formalisables).

Le processus schématique présenté ci-après présente un scénario d'utilisation d'ARLANG dans le cadre de l'implantation d'un aménagement dans un site.



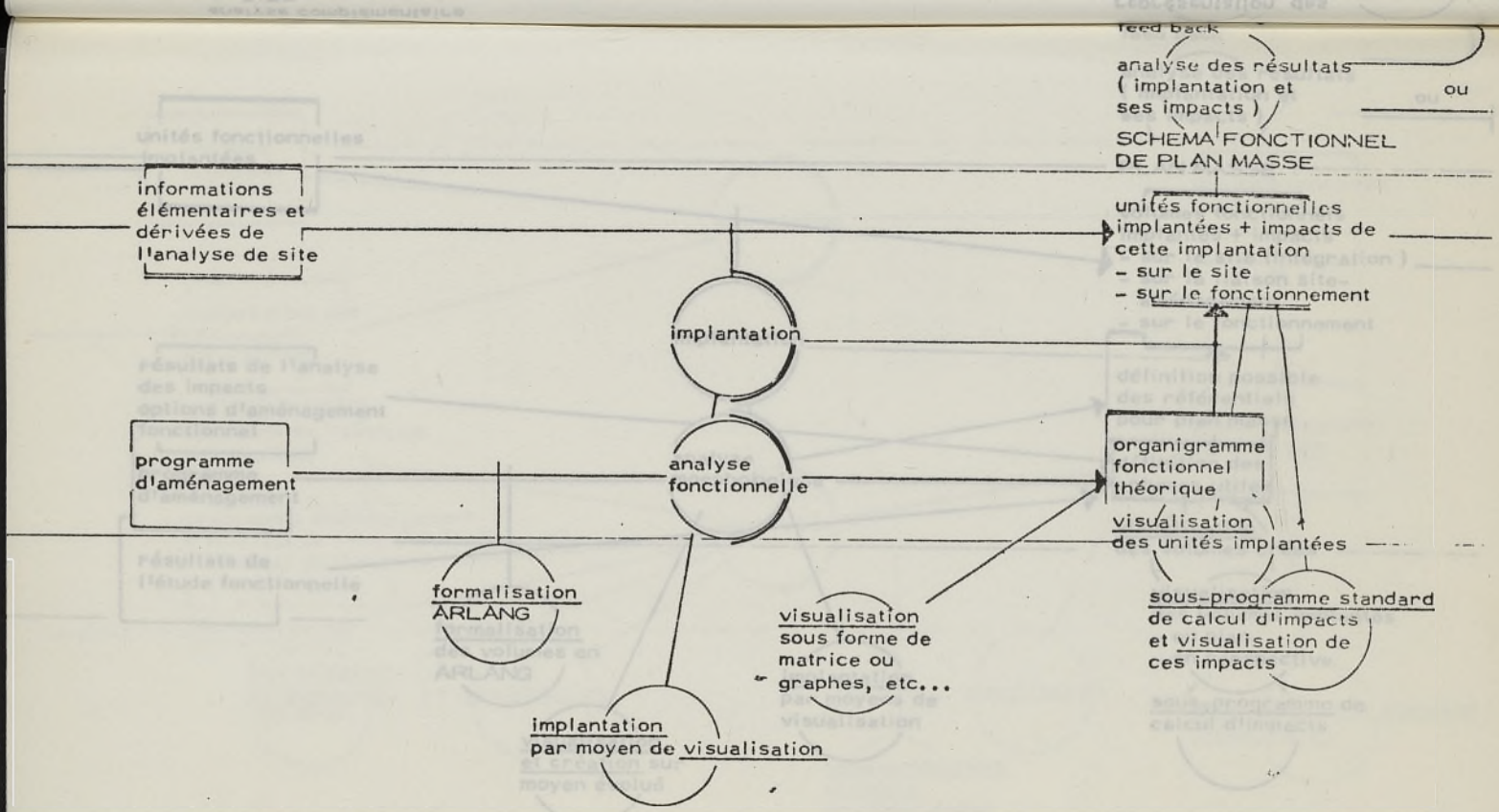
1 - ANALYSE DE SITE

11 recueil des informations



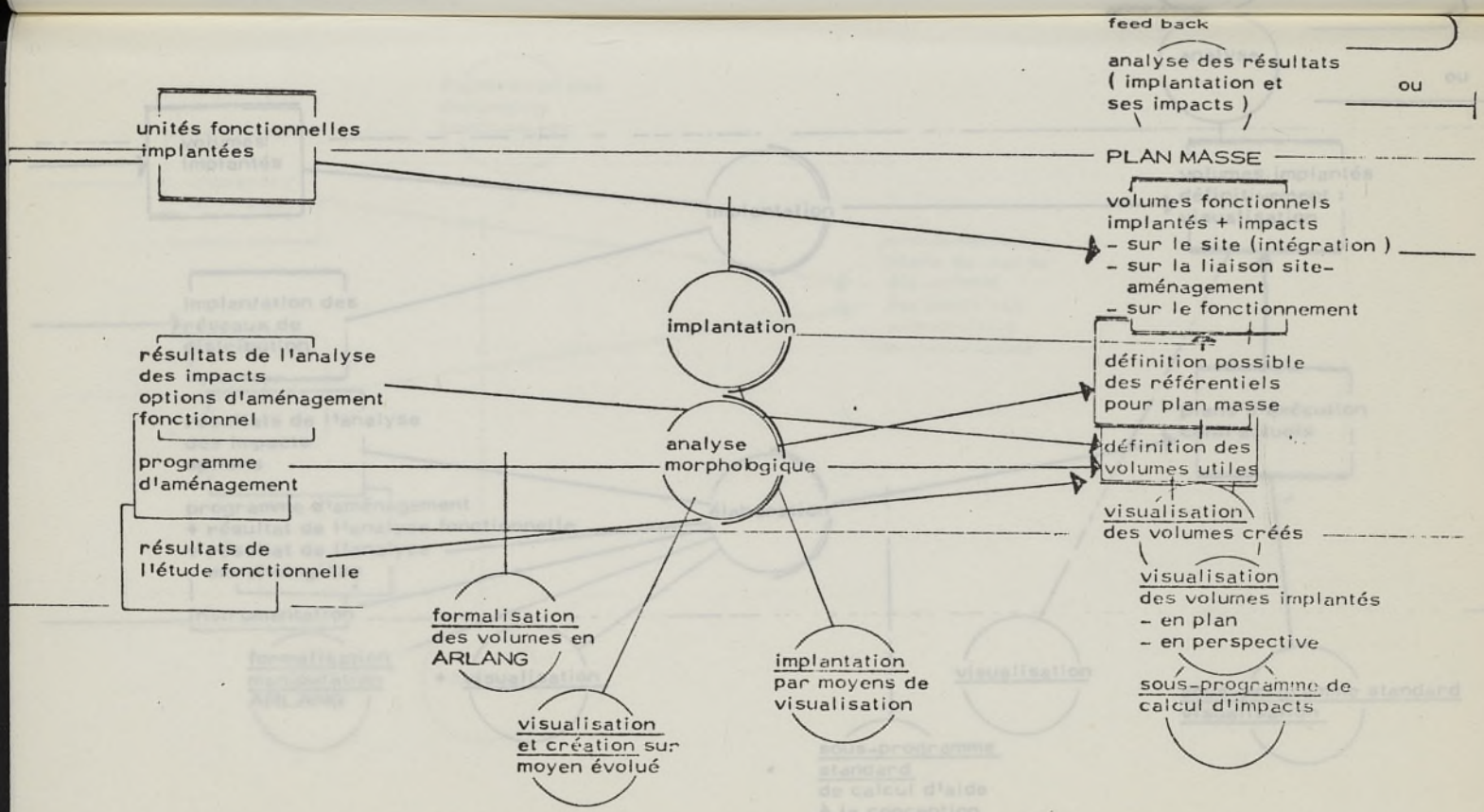
1 - ANALYSE DE SITE

12 classement des informations



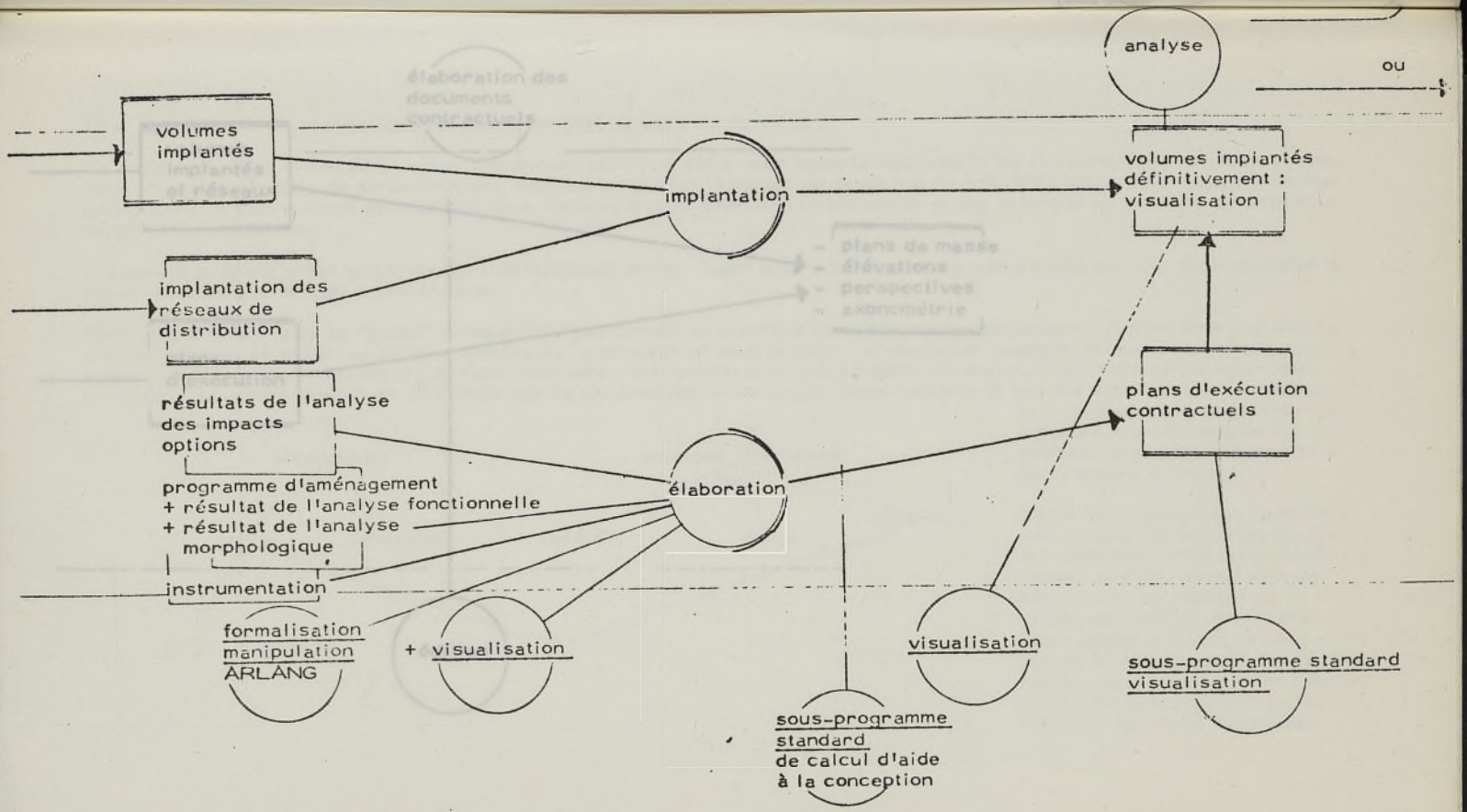
2 - IMPLANTATION

21 fonctionnelle



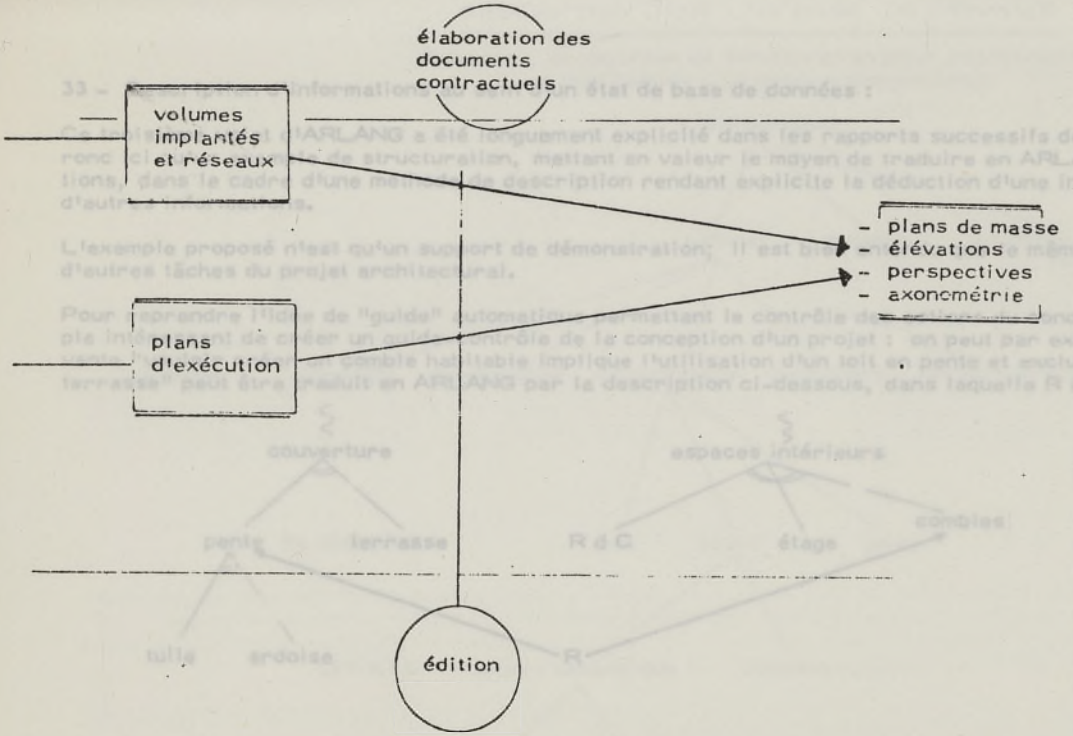
2 - IMPLANTATION

22 des volumes



2 - IMPLANTATION

23 plan masse définitif


élaboration des documents contractuels
volumes implantés et réseaux
plans d'exécution
- plans de masse
- élévations
- perspectives
- axonométrie
édition

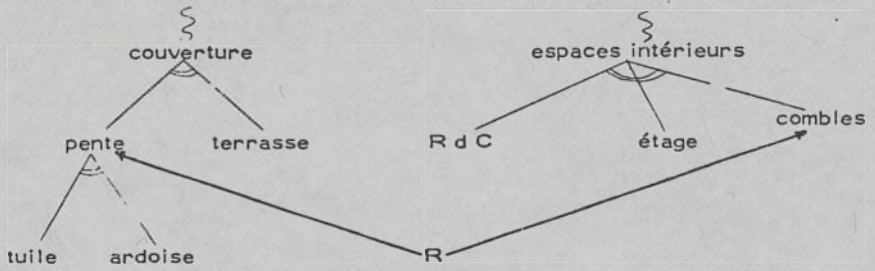
3 - EDITION

33 - Description d'informations au sein d'un état de base de données :

Ce troisième volet d'ARLANG a été longuement explicité dans les rapports successifs de la recherche; nous ne donnerons ici qu'un exemple de structuration, mettant en valeur le moyen de traduire en ARLANG un certain type d'informations, dans le cadre d'une méthode de description rendant explicite la déduction d'une information d'après un ensemble d'autres informations.

L'exemple proposé n'est qu'un support de démonstration; il est bien entendu que le même principe peut être appliqué à d'autres tâches du projet architectural.

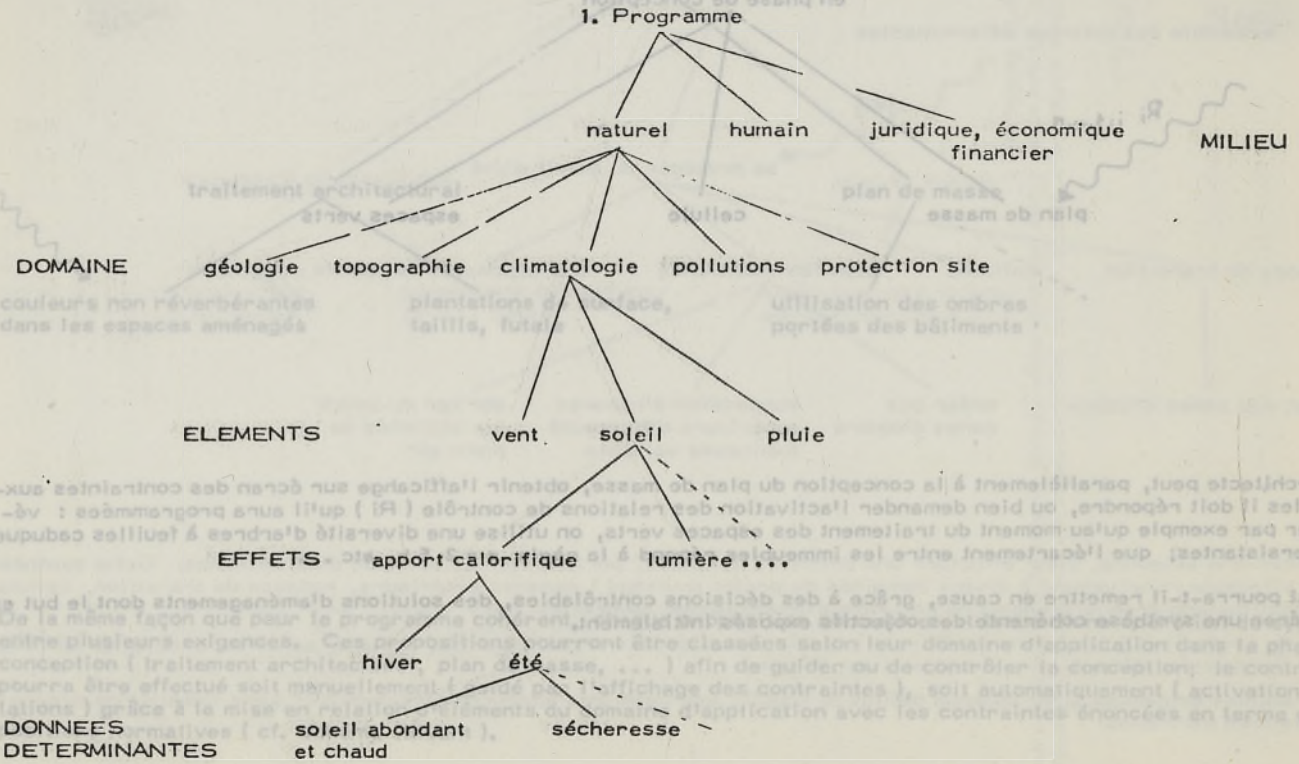
Pour reprendre l'idée de "guide" automatique permettant le contrôle des actions du concepteur, il peut être par exemple intéressant de créer un guide-contrôle de la conception d'un projet : on peut par exemple traduire la phrase suivante "vouloir créer un comble habitable implique l'utilisation d'un toit en pente et exclut donc l'utilisation d'un toit-terrasse" peut être traduit en ARLANG par la description ci-dessous, dans laquelle R est une relation liant couverture : pente et espaces intérieurs : combles, symétriquement et signifiant leur dépendance l'un par rapport à l'autre.



Cette description des relations entre certains éléments du projet constitue une phase nécessairement préliminaire, afin de permettre, au moment de la phase de conception, de contrôler (par exemple quand le concepteur utilise le mot "combles") la pertinence de sa description (ou de guider sa description) en fonction des relations spécifiées.

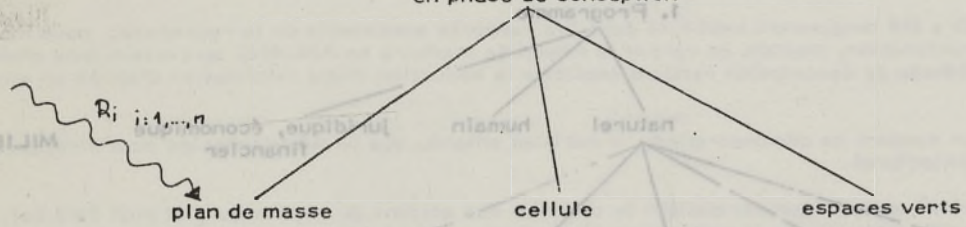
DESCRIPTION DANS UNE BASE DE DONNEES

Exemple : proposition de structuration pour l'élaboration et l'utilisation d'un programme cohérent



Dans cette première phase, les données sont structurées en domaines d'étude spécifiques correspondant à des spécialisation nécessaires au niveau analytique. L'analyse aboutit à l'énoncé d'un ensemble de données déterminantes relatives à chacun des domaines étudiés.

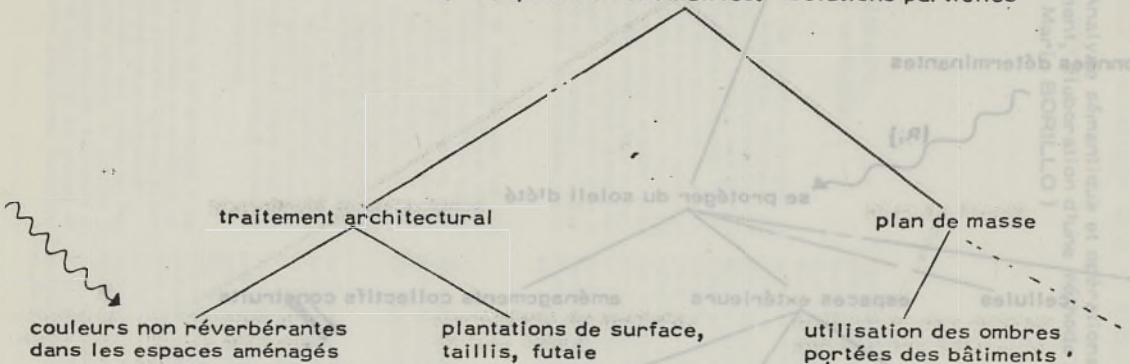
4. Prise en compte des propositions en phase de conception



L'architecte peut, parallèlement à la conception du plan de masse, obtenir l'afficahge sur écran des contraintes auxquelles il doit répondre, ou bien demander l'activation des relations de contrôle (Ri) qu'il aura programmées : vérifier par exemple qu'au moment du traitement des espaces verts, on utilise une diversité d'arbres à feuilles caduques et persistantes; que l'écartement entre les immeubles répond à la règle e = 2,5 h, etc ...

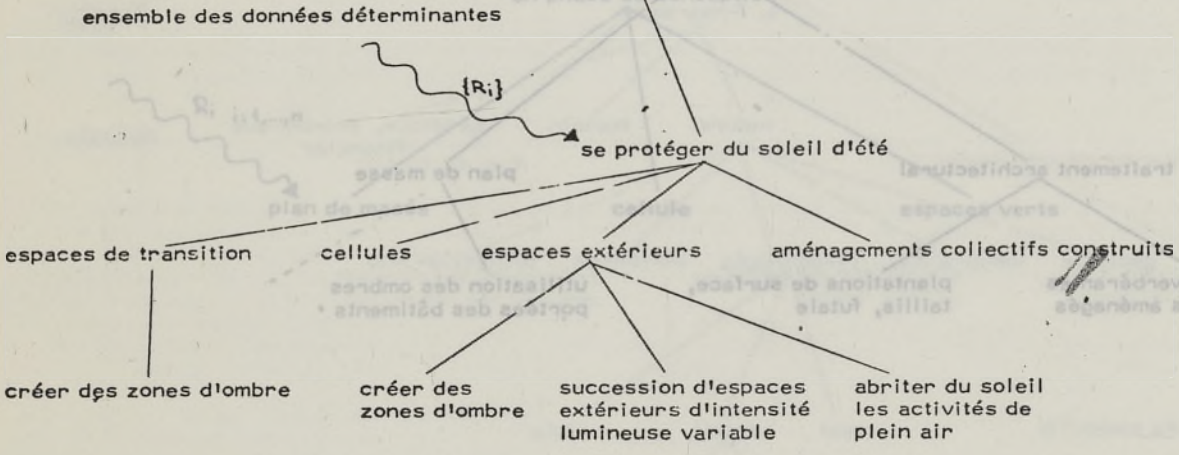
Ainsi pourra-t-il remettre en cause, grâce à des décisions contrôlables, des solutions d'aménagements dont le but est d'opérer une synthèse cohérente des objectifs exposés initialement.

3. Propositions normatives. Solutions partielles



De la même façon que pour le programme cohérent, chaque proposition normative est issue d'une opération de synthèse entre plusieurs exigences. Ces propositions pourront être classées selon leur domaine d'application dans la phase de conception (traitement architectural, plan de masse, ...) afin de guider ou de contrôler la conception; le contrôle pourra être effectué soit manuellement (guidé par l'affichage des contraintes), soit automatiquement (activation de relations) grâce à la mise en relation d'éléments du domaine d'application avec les contraintes énoncées en terme de propositions normatives (cf. schéma suivant).

2. Programme cohérent : exigences



Enoncer une exigence, c'est effectuer une première synthèse, portant sur les données déterminantes. Cette synthèse peut être effectuée relativement à divers domaines du projet envisagé (espaces extérieurs, espaces de transition, cellule ...), et constitue une première approche en terme d'aménagement du projet.

Afin de rendre explicite (donc modifiable) la décision d'énoncé d'une exigence, celle-ci est mise en relation en ARLANG avec les données déterminantes dont elle est déduite.

La liste des exigences ainsi obtenue définit le "quoi", une première approche du "comment" sera l'énoncé de propositions normatives partielles.

ANNEXE 1

Analyse sémantique et opérations logiques dans la conception de l'aménagement, élaboration d'une méthode pour l'étude du Caractère Méditerranéen
(Mario BORILLO)

Analyse sémantique et opérations logiques dans la conception de l'aménagement : élaboration d'une méthode pour l'étude du " caractère méditerranéen ".

MARIO BORILLO

I. - Nécessité d'une méthode.

L'acuité des problèmes auxquels se trouvent aujourd'hui confrontées les techniques de l'aménagement tiennent, pour l'essentiel, à des raisons de deux ordres : (a) fondamentalement, l'importance sans cesse croissante des enjeux, sur les plans économiques, sociaux et par conséquent politiques, enjeux qui placent les questions d'aménagement au centre des préoccupations de tous ; (b) par suite, et en étroite relation avec les acquisitions de la science et de la technologie, la formulation complètement nouvelle des problèmes techniques.

On sait en particulier que l'un des traits caractéristiques de cette modification concerne la quantité et la précision sans cesse croissantes des informations relatives à l'étendue des ressources disponibles comme à la complexité des contraintes à satisfaire. Il est donc naturel que l'on se préoccupe de faire l'inventaire des facteurs qui peuvent se révéler pertinents en même temps que l'on vérifie dans quelle mesure les solutions proposées s'accordent aux conditions du problème. En conséquence, deux types d'objectifs peuvent être isolés dans la conception de l'aménagement.

1. La collecte et le traitement d'une masse d'information en croissance rapide, pour « couvrir » l'étendue du problème et maîtriser sa complexité.
2. L'explicitation des raisons qui justifient toute décision, de manière à rendre communicables, évaluables et de ce fait sujettes à des remises en question les propositions d'aménagement.

Ces objectifs, s'ils sont pris dans leur pleine acception, ne peuvent être atteints que par une approche réellement scientifique mobilisant les moyens techniques appropriés. Autrement dit, la résolution des problèmes de la conception de l'aménagement, tels qu'ils sont maintenant formulés passe par la définition d'une méthodologie. En fait, il faut observer que la diversité des éléments mis en jeu dans tout projet d'aménagement implique que cette méthodologie aura un statut scientifique correspondant à celui des éléments évoqués : sciences de la nature ou « sciences » de l'homme et de la société. C'est dire que si la science de l'aménagement est à venir, sa construction, dès à présent, peut et doit être entreprise.

II. - L'étude du "caractère méditerranéen" de l'habitat en Provence.

Le Groupement des Ateliers Méditerranéens d'Urbanisme, à la demande de l'O.R.E.A.M. Marseille, s'est attaché à l'étude systématique de l'habitat provençal. La nécessité de définir à cette fin une méthode d'analyse s'impose immédiatement lorsque l'exploration des caractéristiques observables et la détermination des caractéristiques souhaitables, pour l'habitat méditerranéen, apparaît dans sa complexité.

La première difficulté porte sur le choix des éléments descriptifs qui seront retenus pour caractériser cet habitat parmi ceux, innombrables, que l'on pourrait isoler, et sur la justification de ce choix.

D'autre part, les observations mises en jeu sont de nature diverse : faits élémentaires, concepts plus ou moins complexes intégrant des faits élémentaires, rationalisations abstraites, etc., d'où la difficulté de traiter directement ces données sans risques de confusion logique.

Enfin, la nécessité de parvenir à des propositions normatives impose que les données préalablement définies et ordonnées fassent l'objet d'une synthèse ; si l'on entend donner à cette dernière un contenu rigoureux, elle devra être envisagée comme le résultat d'un calcul, formalisé ou non.

En même temps qu'une méthode — la trame de la démarche — un certain nombre d'instruments opérationnels se trouvent définis. Leur validité est expérimentée sur un problème précis.

1. — Choix des éléments caractéristiques.

Devant l'abondance et la diversité des domaines d'observation concevables, un premier partage empirique a paru s'imposer, en trois grandes catégories de données, selon la nature du milieu auquel on se réfère pour déterminer l'adéquation spécifique de l'habitat méditerranéen :

A) Milieu naturel, défini par les aspects climatiques, géologiques, etc. de l'environnement ;

B) Milieu technique, défini par la nature des équipements de tous ordres observés ou prévus dans l'aire étudiée ;

C) Milieu politique et social, défini par diverses caractérisations de la population au service de laquelle est destiné l'habitat.

Le nombre des variables mises en jeu dans chacune des trois catégories est tel qu'il a semblé indispensable d'aborder celle-ci de façon séparée, dans la phase initiale de l'étude.

La mise au point de la méthode s'est effectuée à propos du milieu naturel pour la raison évidente que, des trois catégories, c'était celle où les faits observables et les relations entre ces faits pouvaient être depuis sans une très difficile recherche préalable. Il n'en reste pas moins qu'en droit la méthode proposée est tout aussi applicable aux rubriques B et C à la connaissance desquelles elle pourrait apporter une contribution intéressante.
A l'intérieur de chacune des trois catégories, certaines divisions massives se sont également imposées (ex. : dans la catégorie A, la dichotomie Climat-Terre), suivies d'éclatements secondaires en un certain nombre de facettes ou domaines de caractérisation, par une décomposition analytique de l'ensemble des caractérisations retenues.

D'une manière générale, la justification de la démarche analytique est d'abord d'ordre pratique : la séparation en domaines permet d'ordonner la collecte des faits en nombre quasi infini qui peuvent caractériser l'habitat. Quant au fondement théorique du découpage, il est ici comme ailleurs lié à la valeur des constructions qui peuvent en découler, au niveau des synthèses finales et des calculs qui leur sont attachés.

2. — Mise en ordre des éléments de caractérisation.

Une des difficultés principales rencontrées dans ce type d'expériences tient à la diversité des niveaux de conceptualisation où se trouvent les éléments qui interviennent dans chaque domaine, diversité qui constitue un obstacle sérieux à la définition des relations qui existent entre éléments du problème et par suite aux calculs que l'on se propose d'effectuer. On a donc séparé les données élémentaires de l'observation brute (courbes de température, relevé des discontinuités du relief, etc.) rangées au niveau I, des concepts auxquels conduit une interprétation plus ou moins scientifique des données (« inconfort » thermique, « morcellement » du relief, etc.), rangés au niveau II (voir tableaux 1 et 2). Cette première distinction permet de contrôler dans une certaine mesure les fondements objectifs (niveau I) des notions de base (niveau II) qui sous-tendent la réflexion théorique ou l'action pratique, dans les études d'environnement.

Il faut observer que les éléments qui apparaissent dans la rubrique « données » des tableaux établis pour chaque catégorie se rangent en deux classes : les éléments offerts par le milieu (morphologie, ensoleillement, etc.) et ceux qui sont souhaités sur le plan du « confort » de l'homme (qui correspondent aux conditions que l'on cherche à atteindre).

Les concepts induits au niveau II ne relèvent pas, en général, d'une réflexion particulière sur l'aménagement mais plutôt, par construction, du domaine scientifique ou technique (physiologie, géologie, etc.) défini par les données elles-mêmes au niveau I. C'est donc à partir du niveau II que commence une suite d'opérations qui vont maintenant infléchir la démarche synthétique vers des objectifs propres aux études d'aménagement. En particulier, l'interprétation des concepts au niveau II résultera d'un effort de synthèse par lequel on cherche à accorder éléments offerts (milieu) et éléments souhaités (confort) de l'habitat.

Cette interprétation s'inscrira d'abord dans le niveau III, qui est par conséquent la première étape du raisonnement dans la voie d'une conception contrôlée de l'aménagement. En vertu même de la destination pratique de l'étude, on isole un dernier plan de la réflexion, où sont formulées les propositions d'aménagement, fondées sur les interprétations précédentes : c'est le niveau IV de la construction, d'où seront finalement tirés les éléments de la définition normative cherchée, quant aux caractéristiques propres de l'habitat méditerranéen. Ces propositions sont confrontées avec les réponses que la tradition régionale avait elle-même apportées.

L'élaboration de chaque tableau constitue en fait un calcul, au sens large, dont l'objectif est de chercher la solution à un problème particulier défini par la confrontation des deux catégories de données du niveau I. Naturellement, d'une catégorie de l'analyse à l'autre — d'un tableau à l'autre — les éléments sont souvent loin d'être indépendants et il restera donc, pour pouvoir aborder l'étude des problèmes réels d'aménagement, à chercher et à expliciter les relations existant entre les divers tableaux, aux niveaux IV.

III. - Analyse sémantique et calcul dans la conception de l'aménagement.

1. — LA CONSTRUCTION DU RESEAU SEMANTIQUE.

La décomposition de chaque secteur d'observation en un grand nombre de catégories (tableaux) d'éléments découle essentiellement de la nécessité de donner à chaque proposition un contenu aussi précis et aussi objectif que possible. La figure 1 offre un exemple de la finesse suivant laquelle un domaine d'observation doit être découpé si l'on entend dégager son incidence réelle sur l'aménagement.

Le résultat de cet effort d'analyse est l'élaboration d'un système de termes — l'ensemble des propositions de niveau IV — dans lequel chaque élément a un contenu sémantique qui tend à donner à l'ensemble les propriétés d'un langage scientifique, comparable par exemple à celui de la géologie, et caractérisé en premier lieu

Tableau 1

NIVEAUX	DEFINITIONS
IV Propositions	Caractéristiques de tous ordres (technologie, organisation, etc.) à prendre en compte pour une définition raisonnée de l'aménagement.
III Interpretations	Réflexions sur les données et concepts des niveaux antérieurs (I et II), en termes d'aménagement.
II Concepts	Intégration des données (I) par des opérations diverses (corrélation, groupement, synthèse, etc.), dans la perspective des disciplines scientifiques ou techniques concernées (géologie, physiologie, etc.).
I Données	Faits élémentaires intéressant l'homme et son milieu retenus en raison de leur relation possible avec les spécifications de l'habitat.

éléments souhaités — éléments offerts

Tableau 2

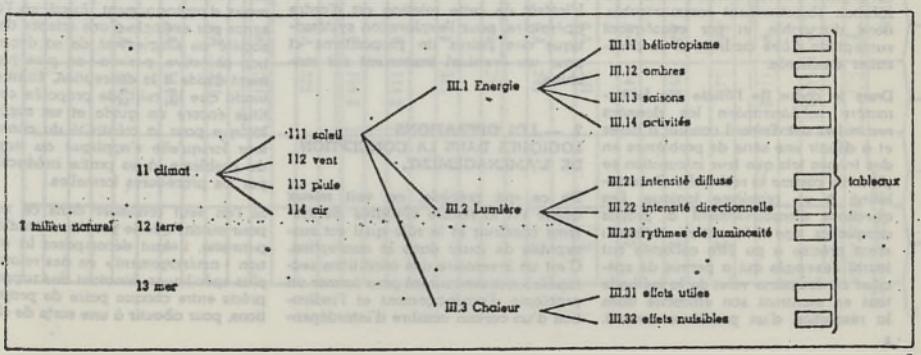
TABLEAU DE RAISONNEMENT A PARTIR DES DONNEES ELEMENTAIRES

Vers les tableaux de synthèse ▲

IV	PROPOSITIONS NORMATIVES	• Notions d'espace de transition. • Mobilité et réglage des occultations (Habitation) Généralisée. • Présence de végétations. • Succession d'espaces extérieurs à intensités lumineuses différentes. • Lieux intérieurs de luminosité différente.	— persiennes (variations diurnes) — plantations (variations saisonnières) — espaces - sas : plantations pergolas porches...	REPONSES DE LA TRADITION	notes
III	INTERPRETATION	Organisation d'un habitat permettant ombre et occultation. Organisation d'un habitat permettant de créer des variations.			
II	CONCEPTS	Longue période d'exposition (> 4 h) inconfortable en été (120 jours). La continuité de luminosité (lumière), surtout en été n'apporte pas les variations nécessaires.			
I	DONNEES DU CONFORT	— inconfort d'une longue (> 4 h) exposition à forte intensité > 40.000 lux pour une activité quotidienne — le besoin de variations de la lumière pour l'homme. CONFORT	Durée (c. s. table) du jour : 8 h 55 en hiver 15 h 20 en été dont 60 % du jour à forte intensité > 40.000 lux. 75 jours d'insolation continue. CLIMAT	DU MILIEU	Rythme de luminosité 111.23 Lumière 111.2 Ensoleillement 111 Climat 11

CONFORT DE L'HOMME DANS UN HABITAT ADAPTE AU MILIEU NATUREL

Fig. 1. — Découpage d'un domaine d'observation : le milieu naturel



par la recherche de rapports biunivoques entre le signifiant (les propositions d'aménagement) et le signifié (la nature réelle des faits d'aménagement correspondants) ; la mise en ordre proposée plus haut n'est rien d'autre que la définition d'une procédure permettant d'établir systématiquement les rapports entre les faits et les termes.

Il s'en déduit, dans la mesure où les rapports des faits entre eux sont connus, ou explorables, qu'il devient possible d'établir sur des bases précises les rapports des termes entre eux.

L'ensemble des réponses locales peut alors être doté d'une organisation sémantique qui indique l'interdépendance des éléments locaux.

On sait que les difficultés dans la conception de l'aménagement tiennent pour l'essentiel à la nécessité d'intégrer pas à pas un très grand nombre de solutions partielles, intégration qui sera évidemment difficile à réaliser, voire impossible, et en tout cas hasardeuse dans ses résultats si les liens de dépendance des propositions ne sont nulle part définis ; on conçoit donc que le fait de disposer du réseau sémantique qui articule les différents termes les uns par rapport aux autres soit d'un intérêt capital. En effet, puisqu'elles ont été définies et explicitées, on peut maintenant essayer de représenter conventionnellement les relations de manière à se donner les moyens d'effectuer un véritable raisonnement sur l'ensemble des propositions ou, si l'on préfère, un calcul. Les avantages seront de deux ordres :

(a) sur le plan opérationnel, une approche systématique de la synthèse des solutions partielles deviendra possible ; à la limite, cette synthèse sera justiciable d'opérations formelles, de calculs logico-mathématiques portant sur les propositions, effectués si nécessaire à l'aide d'un ordinateur ; (b) en ce qui concerne l'aménagement, la méthodologie de la conception ainsi esquissée garantira que la genèse de toute solution, dans la mesure où elle est explicitée et raisonnée, rendra cette solution objectivement communicable, donc discutable, et par conséquent susceptible d'être modifiée et en particulier améliorée.

Dans le cadre de l'étude sur le caractère méditerranéen les données recueillies ont d'abord conduit à isoler et à définir une série de problèmes en des termes tels que leur intégration se présente comme la recherche d'une solution à un problème logique. Les questions d'ensoleillement à propos desquelles une information suffisamment précise a pu être collectée ont fourni l'exemple qui a permis de spécifier ce deuxième volet de la méthode tout en montrant son efficacité dans la résolution d'un problème concret.

Quatre relations distinctes définissent les types de rapports qui ont été relevés et explicités entre les propositions (niveaux III et IV des tableaux) prises deux à deux, exclusivement dans leur signification en termes d'aménagement.

1. **Inclusion :** lorsque le signifié de l'un des énoncés est complètement contenu dans le signifié de l'autre.

Exemple : la proposition 111.132 (un habitat protégeant du soleil d'été) est incluse dans la proposition 111.221 (un habitat protégeant du soleil selon la saison).

2. **Exclusion :** lorsque le signifié de l'un des énoncés s'oppose au signifié de l'autre.

Exemple : la proposition 331 (nécessité des arbres en bordure des voies de communication terrestres du point de vue de l'ensoleillement) et la proposition 332 (nécessité de l'absence des arbres en bordure des voies de communication terrestres du point de vue des accidents) s'excluent mutuellement.

3. **Aménagement :** lorsqu'une relation relevant de l'aménagement s'établit entre les deux termes, et que cette relation ne peut être aussi précisément spécifiée que (1) ou (2).

Exemple : il est nécessaire de procéder à un aménagement pour que 111.31 (habitat recevant le soleil aux heures où son apport calorique correspond aux places de centre) et 111.14 (les lieux de séjour prolongé doivent être protégés du soleil de mai à août) soient simultanément satisfaits.

4. **Indifférence :** c'est, par définition, la relation qui s'établit entre deux termes pour lesquels on ne peut reconnaître aucune des trois premières relations (il faut rappeler que l'on traite exclusivement de rapports précis d'aménagement) ; ainsi, entre 111.14 (voir ci-dessus) et 111.222 (nécessité d'un contrôle des surfaces réverbérantes).

L'intérêt de cette relation est d'ordre opératoire, pour l'exploration systématique des paires de propositions et pour un éventuel traitement sur machine.

2. — LES OPERATIONS LOGIQUES DANS LA CONCEPTION DE L'AMENAGEMENT.

De ce qui précède, on voit mieux quelle est la nature véritable du système construit et le rôle qu'il est susceptible de jouer dans la conception. C'est un inventaire des conditions sectorielles qui concourent pour former un problème d'aménagement et l'indication d'un certain nombre d'interdépendances qui, lorsqu'elles ont été notées, sont mises à profit dans la recherche d'une solution d'ensemble. La contribution des relations est de deux ordres :

— dans le sens d'une simplification, en permettant de grouper des conditions, équivalentes ou incluses les unes dans les autres ; ou encore pour l'élimination de certaines d'entre elles, lorsqu'il y a exclusion en fonction d'ordres explicites de priorité, plus généralement en autorisant des opérations logiques qui sont autant d'étapes formelles dans la résolution du problème ;

— par la localisation méthodique des problèmes précis, signalés par la relation « aménagement » ; autrement dit, par la définition exacte du problème général (Tableau 3).

On observera cependant que si le résultat est une formulation nouvelle du problème, qui est déjà en soi une contribution à sa solution au terme d'une double démarche analytique et intégrative, il n'en reste pas moins que nous sommes loin des procédés algorithmiques de conception de l'aménagement tels qu'ils sont mis en œuvre dans le projet « URBAN V » ou dans les travaux du groupe d'architectes et d'informaticiens du West-Sussex County Council. Ces expériences supposent nécessairement la construction de véritables métalangages scientifiques couvrant la totalité du champ dans lequel s'insère le problème (collecte systématique des ressources élémentaires et des solutions locales, lexique des termes utilisés, inventaire des relations qui organisent l'ensemble des éléments et en particulier logique des propositions qui définissent les problèmes ou les solutions, etc.). A ce prix, la génération de solutions globales par le seul calcul est effectivement réalisable. On voit bien, par la nature des données qu'ils font intervenir et qui appartiennent souvent à des domaines dont le statut scientifique est mal assuré, que cette démarche n'est que très exceptionnellement susceptible d'aboutir pour les problèmes normaux d'aménagement. Aussi, en l'absence par exemple d'une science de la société ou simplement de sa description objective, parlera-t-on plus justement d'aide à la conception, étant entendu que la méthode proposée constitue encore un guide et un support logique pour la créativité du concepteur lorsqu'elle s'applique au noyau du problème, à sa partie irréductible par les procédures formelles.

Si l'on veut continuer dans ce sens, pour autant que le problème étudié le permette, il faut décomposer la relation « aménagement » en des relations plus spécifiques dénotant des rapports précis entre chaque paire de propositions, pour aboutir à une sorte de clas-

Tableau 3

MISES EN RELATIONS

inclusion)
exclusion X
aménagement Δ
indifférence O

ANNEXE 2

Exemple d'analyse et de synthèse pour l'étude d'un programme architectural

			Exigences	Propositions normatives solutions partielles
Milieu naturel	- Espaces extérieurs structurants adaptés au climat méditerranéen en Provence	- Vent N-NO le plus violent et le plus fréquent - Vent E-SE	- Nécessité d'abriter les espaces extérieurs où les piétons stationnent des vents dominants et particulièrement du N-NO - Nécessité d'atténuer la vitesse d'écoulement de l'air au voisinage du sol dans les passages piétonniers et les aires de stationnement des véhicules	- Continuité des bâtiments déterminant des enclos privilégiés - Barrières à vent de jonction des bâtiments et au sol - Epannelage régulier des immeubles - Acrotères perforées formant barrières à vent sur face NORD - Plantations en masse et non en rideaux linéaires
		- Soleil d'hiver	- Nécessité de capter le soleil d'hiver	- Exposition SUD $\pm$ 22° - Masques au SUD : Dimensions des espaces extérieurs L = 2,5 H au minimum (sur ligne NORD-SUD) arbres à feuilles caduques
		- Soleil d'été abondant et chaud	- Nécessité de construire avec l'intention de créer des zones d'ombre - Nécessité d'une succession d'espaces extérieurs à intensités lumineuses différentes - Nécessité d'espaces extérieurs étroits et ombragés - Nécessité d'orienter les circulations et les cheminements - Nécessité d'abriter du soleil les activités publiques de plein air	- Plantations à feuilles caduques - Volumes des plantations dosés - Utilisation du relief existant ou créé - Circulations et parkings semi-enterrés - Parasols, tentes, arbres à feuilles caduques

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu naturel (suite)	- Espaces extérieurs structurants adaptés au climat méditerranéen en Provence (suite)	- Réverbération	- Nécessité d'éviter les réverbérations du sol été-hiver	- Surfaces du sol des espaces extérieurs non réverbérantes (couleurs sombres, plantations : ombres ...)
		- Chaleur sécheresse	- Nécessité de maintenir des zones de fraîcheur	- Espaliers ou talus $\leqslant$ 5 %
			- Stocker l'eau de ruissellement pour les plantations	- Eponges de rétention sous les plantations
		- Pluie	- Nécessité de ralentir le ruissellement des eaux pour éviter le ravinement	- Plateformes successives plantées pratiquement plates et en espaliers
		- Pollutions . poussière et vent	- Nécessité de limiter la formation et la dispersion des poussières	- Zones de terrenue abritées des vents (revêtements solides, plantations couvrantes)
		- Bruits . bruits extérieurs de la route nationale	- Nécessité de ne pas exposer les habitants dans leurs logements	- Talus planté le long de la nationale et "non altus tolendi" à partir de ce talus
		. bruits des voitures	- Nécessité d'isoler habitations	- Espaces tampons (plantations)
		. bruits des voitures	"	- Parkings de desserte semi-enterrés
		. bruits aériens d'autre origine	"	- Sols non réverbérants pour les espaces extérieurs

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu humain	- Espaces extérieurs structurants redonnant une signification à ces espaces favorisant une vie sociale privative et collective satisfaisante	- Piétonnier support des équipements collectifs . Réseaux de piétonniers publics, communs, privatifs à organiser à partir de territoires à spatialiser	- Nécessité d'organiser l'espace à partir d'un cheminement piétonnier principal indiqué sur le PAZ - Nécessité de différencier les espaces suivant les notions de territoires pour en dégager les mailles cohérentes conséquences des inter-relations - cf annexe "Recherche de structuration de l'espace en fonction des notions instinctives"	- Organisation spatiale orientée vers une neutralité au plan de l'incitation et vers la plus large ouverture au plan des vécus possibles - Chaque territoire (tout espace vécu) ressenti comme privatif, commun, ou public, affecté d'une possibilité d'identification indiscutable - Chaque territoire ne se chevauche pas
		- Territoire public Eviter tout rassemblement public hors des espaces publics	- Nécessité d'avoir un territoire fonctionnellement très précis . reconnu sans hésitation . spatialement bien limité . sécurisant . favorisant formes de coopération . demeurant hors d'atteinte de toute appropriation individuelle	- Territoire public englobe les territoires communs et privatifs plus des espaces véritablement publics affectés en indivision anonyme - Territoire public ressenti comme extérieur aux territoires communs et privatifs - Territoire clairement limité - Territoire public défini par des éléments actifs publics attractifs ou légaux

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu humain (suite)	- Espaces extérieurs structurants redonnant une signification à ces espaces favorisant une vie sociale privative et collective satisfaisante (suite)	- Voies de cheminement: Permettre à des piétons ou à des véhicules avec des piétons à l'intérieur de se déplacer, de stationner et de se rencontrer à l'intérieur de territoires communs et privatifs	- Nécessité d'organiser les cheminements et les points de stationnement et de rencontre à l'intérieur des tissus urbains communs et privés pour passer sans heurt : du public au commun du public au privatif du commun au commun du commun au privatif du privatif au privatif	- Les voies communes ne traversent pas les territoires publics ou privatifs - Les voies privatives ne traversent pas les territoires communs ou publics - Le cheminement public ou commun se fait le long de territoires privatifs - Les territoires de promenade sont entièrement protégés - Les parkings communs sont situés dans des espaces communs neutralisés sous surveillance commune - Les garages privatifs sont situés dans des espaces privatifs (niveau bas semi-enterré des bâtiments) sous surveillance privative
	qui ne soient plus conditionnés par une trame circulatoire a priori	- Circulations, cheminements et stationnement des véhicules, conséquences de la structuration des espaces extérieurs privatifs, communs et publics	- Nécessité d'adapter la trame de circulation, de cheminement et de stationnement des véhicules a posteriori à l'organisation des territoires communs et privatifs	- Les voies de circulation publique sont données par le PAZ - Les cheminements communs reliant le public au privatif sont prévus en façades Nord (à l'ombre inévitable des bâtiments) en territoires neutralisés et semi-enterrés en profitant du relief ou des déblais laissés sur place

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu humain (suite)	- Espaces extérieurs structurants redonnant une signification à ces espaces favorisant une vie sociale privative et collective satisfaisante (suite)	- Territoire privatif (suite)	- Nécessité de protéger ces territoires des atteintes de certaines nuisances : vue, odeurs, bruits ...	- Création de masques de verdure; disposition des bâtiments pour limiter les vues directes, écrans aux bruits et environnement non réverbérants
		- Liaisons et relations entre les territoires . desserte . stationnement . rassemblement	- cf annexe "Recherche de structuration de l'espace en fonction des notions instinctives"	
		- Voies de circulation : Permettre à des véhicules avec des piétons à l'intérieur de se déplacer hors des territoires communs ou privatifs et de stationner à l'intérieur du territoire public	- Nécessité d'organiser les circulations à l'extérieur des tissus urbains communs et privatifs dans des territoires publics - Nécessité d'avoir des voies de circulation et des aires de stationnement fonctionnellement précisées . reconnues sans hésitation . spatialement limitées . favorisant les échanges	- Voie publique ne traverse pas de territoire commun ou privatif - Accès public réalisé face aux territoires publics - Aires de stationnement situées en zones neutralisées sous surveillance
		- Voies de circulation piétons Permettre à des piétons de se déplacer hors des territoires communs ou privatifs et de se rassembler à l'intérieur de territoires publics	- Nécessité d'avoir des voies de circulation-piétons et des points de rassemblement fonctionnellement précisés . reconnues sans hésitation . spatialement bien limités . favorisant les échanges	- Voie publique ne traverse pas de territoire commun ou privatif - Accès public réalisé face aux territoires publics - Aires de rassemblement situées dans les équipements publics

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu humain (suite)	- Espaces extérieurs structurants redonnant une signification à ces espaces favorisant une vie sociale privative et collective satisfaisante (suite)	- Territoire commun Interdire à tout individu extérieur au groupe usage et accès sauf accès par des passages précis jusqu'aux individus recherchés	- Nécessité de définir les territoires communs . reconnus comme situés entre territoires publics et privatifs . considérés comme privatifs vus de l'extérieur et comme communs vus de l'intérieur - Nécessité de préciser les passages en leur ôtant tout caractère public - Nécessité de maintenir ces territoires hors d'une appropriation abusive d'une personne d'une fraction de groupe	- Territoire commun englobe les territoires privatifs plus un espace véritablement commun affecté en indivision, mais non public - Territoire clairement limité et borné - Territoire commun défini par des éléments actifs communs et collectifs - Passage d'un territoire commun à un territoire privatif comportant des paliers de protection permettant à celui qui sort ou qui entre de signifier ses intentions et son itinéraire
		- Territoire privatif Interdire à tout individu extérieur au privatif usage et accès sauf accès par des passages précis et sans heurt de quelques uns	- Nécessité de définir les territoires privatifs extrêmement variables . reconnus comme des culs de sac que l'on ne traverse pas . spatialement limités avec une extrême précision - Nécessité que les points d'entrée (et de sortie) soient immédiatement reconnus comme normaux - Nécessité de maintenir ces territoires hors d'une appropriation abusive d'une personne ou d'une fraction de groupe	- Territoire privatif - avant tout privatif - considéré comme contenu dans les territoires commun ou public, par encerclement et non par recouvrement - Territoire clairement limité et clos - Vis à vis neutralisé par interposition d'un territoire commun ou public - Passage d'un territoire commun à un territoire privatif comportant des paliers de protection permettant à celui qui sort ou qui entre de signifier ses intentions et son itinéraire

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions particlles
Milieu hurrain (suite)	- Espaces extérieurs structurants redonnant une signification à ces espaces favorisant une vie sociale privative et collective satisfaisante (suite)	- Circulation, cheminements et stationnement des véhicules (suite)	- Nécessité d'adapter la trame de circulation, de cheminement et de stationnement des véhicules a posteriori à l'organisation des territoires communs et privatifs (suite)	- Les aires de stationnement sont situées le long des cheminements des véhicules et semi-enterrés également - Le complément des parkings nécessaires est situé globalement à l'entrée des espaces communs près de la garderie d'enfants ou sous les accès à la maternelle
		- Equipements et locaux à caractère polyvalent (privatifs, communs, publics) en position d'échanges	- Nécessité d'intégration des équipements collectifs au quartier	- Etalement des écoles dans une rue - cour de récréation dans un territoire public enserré par des territoires communs ou privatifs - La rue passe à travers l'école - Eléments d'accueil maternelle ou salle ouverte de l'école primaire à usage polyvalent (coin de détente des personnes âgées ou lieu de formation pour les adultes) - Restaurant d'enfants placé en charnière entre le primaire et le CES le long du piétonnier principal

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu naturel	- Prolongement de l'habitation sur l'extérieur adapté au climat méditerranéen en Provence	- Vent N-NO le plus violent et le plus fréquent et vent S-SE	- Nécessité d'abriter ces espaces de prolongement des vents dominants	- Espaces implantés à l'abri des bâtiments-écrans, ou des masses de plantations
		- Soleil d'hiver	- Nécessité de capter le soleil d'hiver	- Exposition SUD $\pm$ 22°
		- Soleil d'été abondant et chaud	- Nécessité de créer des zones d'ombre	- Enclos privilégiés avec plantations à feuilles caduques
		- Réverbération	- Nécessité d'éviter les réverbérations du sol été-hiver	- Surfaces du sol non réverbérantes
		- Chaleur-sécheresse	- Nécessité de créer des zones de fraîcheur	- Zones d'ombre ou arrosables
Milieu humain	- Prolongement de l'habitation sur l'extérieur	- Création des espaces de transition du commun au privatif	- Nécessité d'aménager des espaces tampons entre le privatif et le commun	- Coursives, monte charges (coursive verticale) - Cheminements piétonniers
	. redonnant à ces espaces une signification	- Espaces communs favorables aux échanges dans le bâtiment	- Nécessité d'aménager des espaces tampons réduisant les possibilités de conflits	- Cages d'escaliers en entrées multiples remplacées par entrée unique, monte charges, coursives, escaliers de secours extérieurs
	. favorisant une vie sociale satisfaisante	- Surface privative extérieure à la cellule de 15 à 50 m2 réellement utilisable	- Nécessité de donner au plus grand nombre de logements les avantages du logement individuel	- Espaces privatifs $\geqslant$ 50 m2 au sud des habitations individuelles et semi-individuelles à R. de C. - Espaces privatifs de 15 m2 en couvertures terrasses dans les habitations semi-individuelles

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu humain (suite)	- Espaces extérieurs structurants redonnant une signification à ces espaces favorisant une vie sociale privative et collective satisfaisante (suite)	- Variété d'échelles des volumes	- Nécessité de spatialiser l'ensemble en gardant la notion d'échelle humaine - Nécessité d'avoir des volumes variés dans leur nature et leur encombrement rattachés à la perception humaine	- Bâtiments de hauteur faible et progressive - Volumes bâtis prolongés ou masqués par des volumes plantés de hauteurs différentes - Volumes bâtis et plantés orientés suivant des obliques et non des parallèles - Equipements extérieurs (jeux d'enfants, jeux de boules, plantations en masse ou en ligne) traités en éléments restreints à de petites dimensions
Milieu économique administratif et règlementaire	- Espaces extérieurs structurants répondant au réalisme des propositions satisfaisant au binôme qualité-prix satisfaisant en investissement-entretien-gestion	- Economie du projet Construire à moindre coût un habitat de qualité	- Nécessité de supprimer les installations les plus onéreuses - Nécessité d'équiper avec du robuste, du résistant, et d'un entretien économique - Nécessité de plannifier le financement pour une réalisation effective	- Un parking collectif couvert revient à 10 000 fr la place un parking individuel non couvert revient à 2 500 fr la place - Réduction maximale des parkings collectifs couverts - Incorporation maximale de parking-garage individuel en semi-infrastructure des bâtiments - Bilan des investissements à faire pour une réalisation effective
	respectant les règles administratives	- Respect absolu du PAZ	- Nécessité de satisfaire aux règles de la ZAC	

LES ESPACES EXTERIEURS

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions partielles
Milieu humain (suite)	- Espaces extérieurs structurants redonnant une signification à ces espaces favorisant une vie sociale privative et collective satisfaisante (suite)	- Equipements et locaux à caractère polyvalent (privatifs, communs, publics) en position d'échanges (suite)	-Nécessité d'englober les espaces réservés aux activités extérieures et polyvalentes des adultes aux territoires communs et publics	- Locaux résidentiels et espaces d'activités extérieures des adultes placés en charnière entre la maison pour tous et les territoires communs ou privatifs, le long du piétonnier principal - Ateliers de détails, prolongements de la maison pour tous, situés de l'autre côté du piétonnier principal
			- Nécessité de créer des enclos variés pour les jeux et loisirs des enfants	- Enclos des tous petits ensoleillés, protégés, à proximité de la surveillance immédiate de la mère ou de la maîtresse d'école - Enclos plus vaste, aussi protégé et ensoleillé à l'intérieur des territoires communs et publics sous surveillance commune pour les jeunes enfants - Enclos et locaux polyvalents dans les territoires publics principalement pour les adolescents

LES ESPACES DE TRANSITION

Domaine	Programme MAEB	Données déterminantes	Programme cohérent Exigences	Propositions normatives solutions particlles
Milieu économique administratif et règlementaire	- Prolongement de l'habitation sur l'extérieur . répondant au réalisme des propositions . satisfaisant au binôme qualité-prix . satisfaisant en investissement entretien-gestion	- Construire au moindre coût un habitat de qualité	- Nécessité d'apporter des solutions réduisant les dépenses des parties communes et permettant un entretien et une gestion économiques	- Un seul monte charges robuste à un seul niveau et à une seule commande (aller-retour) - Une seule coursive incorporée au bâti, fermée par des éléments préfabriqués standards et traitée comme une rue couverte - Passerelle de liaison entre deux bâtiments aboutissant à un monte charges ou à un escalier de secours, traitée en matériaux robustes

CONCLUSIONS

L'état actuel de notre recherche ne nous permet pas d'énoncer de conclusions précises sur le langage proposé, car nous ne sommes aujourd'hui qu'au tout début de son implémentation; cette implémentation devra être poursuivie pour permettre l'expérimentation d'un sous modèle du langage (1).

A ce jour nous pensons cependant pouvoir juger de la pertinence générale du langage en ce qui concerne les objectifs présentés dans la partie I.

Si, de façon générale, ces objectifs ont été atteints dans leur ensemble, ils l'ont été au détriment de l'un d'eux : réaliser un langage directement accessible par un architecte non spécialiste de l'informatique, car au fur et à mesure de son enrichissement il est devenu de plus en plus complexe, malgré toutes les solutions techniques introduites en vue de simplifier la programmation.

Actuellement le langage proposé paraît être un outil de programmation adapté à la description et à la résolution de problèmes architecturaux, lorsque celle-ci est réalisée par des architectes déjà formés à la programmation, il l'est moins en ce qui concerne son utilisation par des architectes profanes.

Nous avons choisi cette voie car nous pensons que la priorité devait être donnée à un outil sophistiqué au détriment d'une complexité accrue; cette complexité pouvant être réduite ultérieurement par une amélioration du langage portant sur :

- la mise en place d'un système conversationnel d'apprentissage du langage,
- la mise en place de procédures standards tendant à rendre ARLANG "presse-boutons" tout en conservant les possibilités de son extension,
- l'élaboration d'un manuel de programmation simplifié (hiérarchisation des difficultés, illustration par des exemples concrets, etc ...).

En dehors du problème de l'apprentissage de la programmation, va se poser celui, plus fondamental, concernant les modes de description des informations architecturales manipulées au cours du processus de conception architecturale. Une telle recherche devrait aboutir :

- à la mise en place d'un système interactif de conception en architecture, grâce à l'étude des procédures de stimulation de la créativité, des phénomènes d'apprentissage, qui devienne un guide pédagogique d'aide à la conception,
- à la mise en place d'un système de contrôle de pertinence des descriptions proposées par l'architecte relativement à un modèle type.

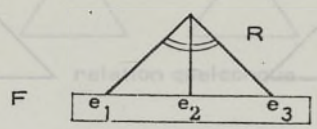
(1) Le caractère hautement sophistiqué de ce langage nous a conduit à envisager dans un premier temps seulement l'implémentation d'un sous modèle. Une extension de la recherche consisterait à écrire l'interpréteur complet du langage.

ANNEXES

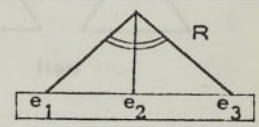
Annexe 1 : Symboles et formalisme

1 - Métalangage ARLANG : notation schématique

- Descripteur élémentaire de type ou :

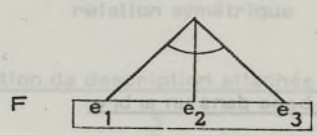


ou

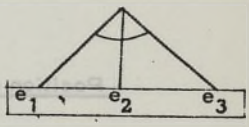


Il est conseillé de noter en minuscules les "valeurs" des noeuds, les identi cateurs étant en majuscules.

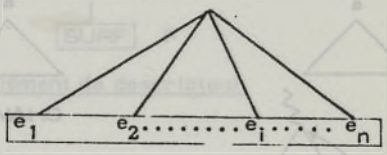
- Descripteur élémentaire de type et :



ou

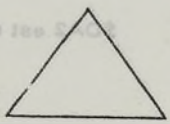


- Descripteur élémentaire de type ou ou et à n éléments



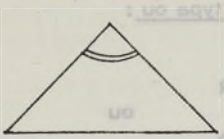
e_i : élément génératif

- Descripteur quelconque

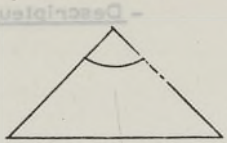


- Descripteur quelconque

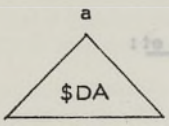
de type ou



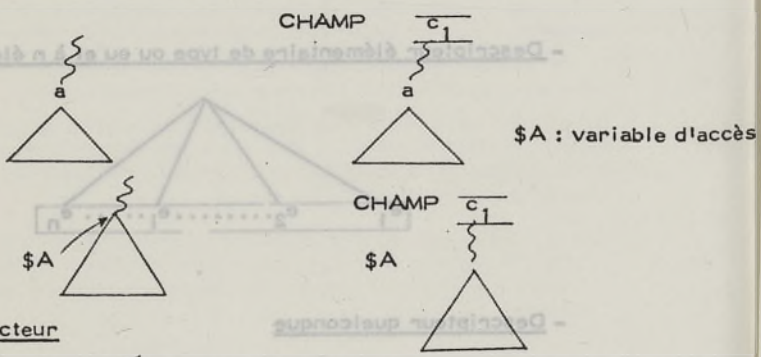
de type et



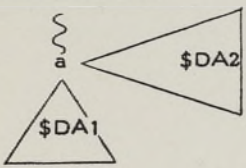
- Descripteur d'un noeud a



- Position d'un élément de base dans un arbre

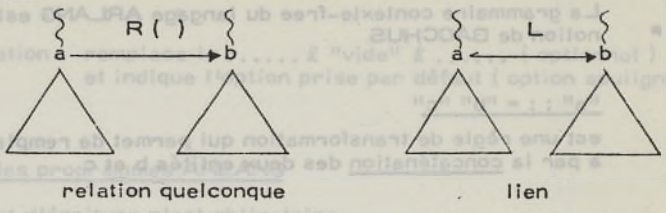


- Mise en facteur

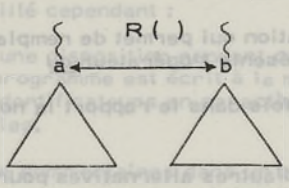


$DA2 est en facteur sur $DA1

- Relations

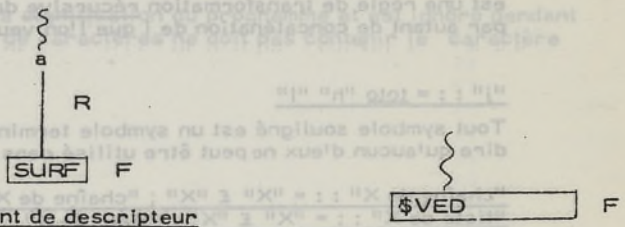


relation quelconque lien



relation symétrique

- Fonction de description attachée à un noeud :



- Variable élément de descripteur

2 - Notation des règles syntaxiques :

La grammaire contexte-free du langage ARLANG est décrite à l'aide de la notion de BACCHUS

"a" : : = "b" "c"

est une règle de transformation qui permet de remplacer l "entité syntaxique" a par la concaténation des deux entités b et c

"d" : : = "e" £ "f" £ "g"

est une règle de transformation qui permet de remplacer d par e ou par f ou par g - Le symbole £ représente l'opérateur ou

nb : nous utilisons quelquefois dans le rapport la notation dérivée :
"d" : : = ... £ f £ ...

Cette notation signifie que d'autres alternatives pour le remplacement de d existent, mais ne sont pas d'intérêt dans le sujet traité dans ce paragraphe.

"h" : : = "i" "h" £ "i"

est une règle de transformation récursive dans laquelle h peut être remplacé par autant de concaténation de i que l'on veut.

"j" : : = toto "h" "l"

Tout symbole souligné est un symbole terminal appelé "mot réservé", c'est-à-dire qu'aucun d'eux ne peut être utilisé dans un autre rôle.

"chaîne de X" : : = "X" £ "X" ; "chaîne de X"
"liste de X" : : = "X" £ "X" ; "liste de X"
"suite de X" : : = "X" £ "X" "suite de X"

X est remplacé dans les règles par certaines entités syntaxiques.

- L'ensemble des règles syntaxiques d'ARLANG pourront être décrites par une arborescence ARLANG dont la racine est l'entité "programme arlang" et les feuilles l'ensemble des symboles terminaux. On pourra ainsi envisager l'appel en mode bureau de cette base de données standards pour obtenir instantanément une règle de transformation donnée.

- Une forme syntaxique plus "parlante" sera proposée ultérieurement, dont les deux symboliques essentiels sont :

$\left\{\begin{matrix}\text{"a"}\\\text{"b"}\\\text{"c"}\end{matrix}\right\}$ signification : remplace le £

$\left[\frac{d}{e}\right]$ signification : remplace le ..... £ "vide" £ ..... (optionnel)
et indique l'option prise par défaut (option soulignée)

3 - Notation des programmes ARLANG

- Aucun format d'écriture n'est obligatoire
- Il est conseillé cependant :

1) d'adopter une disposition rendant compte des imbrications de bloc
2) quand un programme est écrit à la main, de souligner les mots réservés, d'écrire les identificateurs en caractères majuscules, les éléments de famille en minuscules.

- Inclusion de commentaires dans un programme en mode exécutable

Syntaxe :

"commentaire" : : = < "suite de caractères">

Il n'a pas d'effet sur la signification du programme et est ignoré pendant l'exécution. La suite de caractères ne doit pas contenir le caractère >

A.3.

2 - Notation des règles syntaxiques :

La grammaire contexte-free du langage ARLANG est décrite à l'aide de la notion de BACCHUS

$\begin{bmatrix} b \\ a \end{bmatrix}$ signification : remplace le £ "vide" £ ..... £ (optionnel)
et indique l'option prise par défaut (option soulignée)

"a" : : = "b" "c"

est une règle de transformation qui permet de remplacer l'entité syntaxique a par la concaténation des deux entités syntaxiques b et c.

3 - Notation des procédures ARLANG

- Aucun format d'écriture n'est obligatoire
- Il est conseillé cependant :

"d" : : = "e" £ "f" £ "g"

est une règle de transformation qui permet de remplacer d par e ou par f ou par g.

1) d'adopter une disposition rendant aisée la lecture du programme,
2) quand un programme est écrit à la main, de souligner les mots réservés, d'écrire les identificateurs en caractères minuscules,
3) d'adopter une identification mnémotechnique de familles d'identificateurs, par exemple en commençant les noms de variables de type entier par la lettre I, les noms de variables de type réel par la lettre R, etc...

"d" : : = ... £ f £ ...

Le symbole £ signifie un choix entre plusieurs règles de transformation existant, mais ne sont pas d'intérêt dans le sujet traité dans ce paragraphe.

"commentaire" : : = < "suite de caractères" >

"h" : : = "i" "h" £ "j"

est une règle de transformation récursive dans laquelle h peut être remplacé par i suivi de h ou par j. Le symbole £ signifie "ou".

"i" : : = toto "h" "j"

Tout symbole souligné est un symbole terminal appelé "mot réservé", c'est à dire qu'aucun d'eux ne peut être utilisé dans un autre rôle.

"chaîne de X" : : = "X" £ "X" "chaîne de X"
"liste de X" : : = "X" £ "X" , "liste de X"
"suite de X" : : = "X" £ "X" "suite de X"

X est remplacé dans les règles par certaines entités syntaxiques.

- L'ensemble des règles syntaxiques d'ARLANG pourront être décrites par une arborescence ARLANG dont la racine est l'entité "programme arlang" et les feuilles l'ensemble des symboles terminaux. On pourra ainsi envisager l'appel en mode bureau de cette base de données standards pour obtenir instantanément une règle de transformation donnée.

- Une forme syntaxique plus "parlante" sera proposée ultérieurement, dont les deux symboliques essentiels sont :

$\begin{bmatrix} "a" \\ "b" \\ "c" \end{bmatrix}$ signification : remplace le £

s réservés et entités syntaxiques
NEXE 2 :

1 Mots réservés

page du rapport (partie III)
où ils apparaissent

2 Entités syntaxiques

BIBLIOGRAPHIE

- Notes sur la synthèse de la forme
C. ALEXANDER

- A pattern language wich generates multi-services centers
ALEXANDER, ISHIKAWA, SILVERSTEIN

- Méthode systématique d'analyse et de programmation des ouvrages
CIAB

- Notes méthodologiques en architecture et en urbanisme - 3/4 -
Sémiotique de l'espace - Institut de l'Environnement

- Recherche des conditions méthodologiques fondamentales de la collecte et de l'organisation des données dans un champ spécifique des sciences humaines : l'aménagement de l'espace architectural
F. et J.P. CHEYLAN - B. DOMENECH - J. LEMAITRE

- Etude d'outils d'aide à la conception en architecture concernant la recherche et la compréhension de facteurs pertinents guidant le travail de l'architecte pour l'élaboration d'esquisses d'implantation et plan de masse - GAMSAU

- Une ville n'est pas un arbre - in Architecture-Mouvement-Continuité - 1976 - 181 - C. ALEXANDER

- Etude d'une méthodologie d'analyse et de conception d'une base de données - Le schéma de référence
H. TARDIEU, H. HECKENROTH, D. NANCI

-Systèmes interactifs de planification budgétaire. Amplification du raisonnement
D. PASCOT

- Notions sur les grammaires formelles -
GROSS et LENTIN

- Formalisation des notions de machines et programmes
Louis NOLIN

- The art of computer programming - D. E. KNUTH
Vol. 1 Fondamentals Algorithme - Addison Wesby Publishing Compagny

- L'allocation dynamique de la mémoire des ordinateurs - R. EHRMANN
Monographie AFCET

- Les structures de liste et leurs applications - E et A. SIFBON

- Technique récursive en programmation - D. W. BARRON

- Compiler construction for digital computer - D. GRIES

- Article in Revue d'Informatique et de Recherche Opérationnelle - AFCET
  - implantation des arborescences - application au tableau - BAZERQUE n° B3 Décembre 1970
  - analyse syntaxique par cheminement sur un graphe - BLOCH et BRISSEAU - n° B1 Mars 1970

- Le Système Information ARIANE - Information CATED
GAMSAU Bulletin n°1 Vol. 2, pp. 39 à 47

- ARK 2 - Un système d'aide à la conception par ordinateur
K. LEE, GAMSAU Bulletin n°4, Vol. 2 pp. 5 à 10

- Langage formel en allocation spatiale. Un exemple : FOSPLAN
M. BERTHELOT, d'après "FOSPLAN : a formal space planning language"
Chris I. YESSIOS; GAMSAU Bulletin n°1, Vol. 3 pp. 27 à 43

- L'informatique dans la conception de l'aménagement
BORILLO, POUX, QUINTRAND, VO DINCH - 1970

- Architecture et informatique : situation en 1973
P. QUINTRAND, Cahier GAMSAU

- EUCLID - Manuel d'utilisation
BRUN et THERON - LIMSI CNRS - 1975

- The architecture machine - Toward a more human environment
N. NEGROPONTE - 1970

- Computer aids to design and architecture - Edited by N. NEGROPONTE - 1976

- La communication dans le processus de la construction - Classification et codification pour l'utilisation des ordinateurs

- Le courant d'information dans le processus de la construction - CIB

- Données informatiques et conception en architecture - 1971, IRIA

- Analyse des données en architecture et en urbanisme - Institut de l'Environnement

- Analyse sémantique et opérations logiques dans la conception de l'aménagement : élaboration d'une méthode pour l'étude du "caractère méditerranéen"
M. BORILLO - GAMSAU Bulletin n°1, Vol. 1 pp. 3 à 8

- La formulation explicite des données fonctionnelles en architecture
R. BILLON - GAMSAU Bulletin n° 2-3, Vol. 2 pp. 21 à 23

- Note sur les problèmes d'analyse descriptive en architecture
M. BORILLO - GAMSAU Bulletin n°2-3, Vol. 2, pp. 3 à 10

- Problèmes méthodologiques et techniques de la production d'unités volumiques architecturales
R. BILLON, A. GUENOCHE, J. VIRBEL
GAMSAU Bulletin n°1, Vol. 3 pp. 9 à 20