Pour une généralisation des systèmes C.A.O : approche et applications

Bohuslav David

THESE

présentée à

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

POUR OBTENIR LE GRADE DE
DOCTEUR DE 3ème CYCLE
INFORMATIQUE

Bohuslav DAVID

POUR UNE GENERALISATION DES SYSTEMES C.A.O.

APPROCHE ET APPLICATIONS

Thèse soutenue le 25 octobre 1975 devant la Commission d'Examen

Président : J. KUNTZMANN

Examinateurs : H. GALLAIRE
: M. GRIFFITHS
: J. MERMET
: J.C. SABONNADIERE

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

M. Michel SOUTIF — Présidents — M. Louis NEEL

M. Gabriel CAU — Vice-Présidents — MM. Lucien BONNETAIN, Jean BENOIT

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BEZES Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologie
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-Rhino-Laryngologique
	CHATEAU Robert	Thérapeutique (Neurologie)
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique
	CRAYA Antoine	Mécanique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBERMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumo-Phtisiologie
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée

MM.	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DRUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de dermatologie et syphiligraphie
	FAU René	Clinique neuro-psychiatrique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques pures
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Mathématiques appliquées
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique Générale
	KLEIN Joseph	Mathématiques pures
	KOSZUL Jean-Louis	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques pures
	MALGRANGE Bernard	Mathématiques pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histclogie
	MOUSSA André	Chimie nucléaire
	MULLER Jean-Michel	Thérapeutique (néphrologie)
	NEEL Louis	Physique du solide
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-chirurgie
	SEIGNEURIN Raymond	Microbiologie et hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	CHEEKE John	Thermodynamique
	COPPENS Philip	Physique
	CORCOS Gilles	Mécanique
	CRABBE Pierre	CERMO
	GILLESPIE John	I.S.N.
	ROCKAFELLAR Ralph	Mathématiques appliquées

PROFESSEURS SANS CHAIRE

Mlle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
	BERTRANDIAS Jean-Paul	Mathématiques pures
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
Mme	BONNIER Jane	Chimie générale
MM.	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique
	CONTE René	Physique
	DEPASSEL Roger	Mécanique des fluides
	GAUTHIER Yves	Sciences biologiques
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Méd. Préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques pures
	JULLIEN Pierre	Mathématiques appliquées
Mme	KAHANE Josette	Physique
MM.	KUHN Gérard	Physique
	LOISEAUX Jean	Physique nucléaire
	LUU-DUC-Cuong	Chimie organique
	MAYNARD Roger	Physique du solide
	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	PFISTER Jean-Claude	Physique du solide
Mlle	PIERY Yvette	Physiologie animale
MM.	RAYNAUD Hervé	Mathématiques appliquées
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	AMBLARD Pierre	Dermatologie
	ARMAND Gilbert	Géographie
	ARMAND Yves	Chimie
	BARGE Michel	Neurochirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamique
M.	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B)
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DELOBEL Claude	M.I.A.G.
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FONTAINE Jean-Marc	Mathématiques pures
	GAUTIER Robert	Chirurgie générale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GRIFFITHS Michaël	Mathématiques appliquées
	GROS Yves	Physique (stag.)
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	KOLODIE Lucien	Hématologie
	KRAKOWIAK Sacha	Mathématiques appliquées
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LEROY Philippe	Mathématiques
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MARECHAL Jean	Mécanique
	MARTIN-BOUYER Michel	Chimie (CUS)
	MICHOULIER Jean	Physique (IUT A)
Mme	MINIER Colette	Physique
MM.	NEGRE Robert	Mécanique
	NEMOZ Alain	Thermodynamique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PERRET Jean	Neurologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RAMBAUD Pierre	Pédiatrie
Mme	RENAUDET Jacqueline	Bactériologie
MM.	ROBERT Jean-Bernard	Chimie-Physique

MM.	ROMIER Guy	Mathématiques (IUT B)
	SHOM Jean-Claude	Chimie générale
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VROUSOS Constantin	Radiologie

MAITRES DE CONFERENCES ASSOCIES

MM.	COLE Antony	Sciences nucléaires
	FORELL César	Mécanique
	MOORSANI Kishin	Physique

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

MM.	BOST Michel	Pédiatrie
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	FAURE Gilbert	Urologie
	MALLION Jean-Michel	Médecine du travail
	ROCHAT Jacques	Hygiène et hydrologie

Fait à Saint Martin d'Hères, OCTOBRE 1974.

"MEMBRES DU CORPS ENSEIGNANT DE L'I.N.P.G."

PROFESSEURS TITULAIRES

MM. BENOIT Jean	Radioélectricité
BESSON Jean	ELectrochimie
BONNETAIN Lucien	Chimie Minérale
BONNIER Etienne	Electrochimie, Electrométallurgie
BRISSONNEAU Pierre	Physique du solide
BUYLE-BODIN Maurice	ELectronique
COUMES André	Radioélectricité
FELICI Noël	Electrostatique
PAUTHENET René	Physique du solide
PERRET René	Servomécanismes
SANTON Lucien	Mécanique
SILBER Robert	Mécanique des fluides

PROFESSEUR ASSOCIE

M. BOUDOURIS Georges	Radioélectricité

PROFESSEURS SANS CHAIRE

MM. BLIMAN Samuel	Electronique
BLOCH Daniel	Physique du solide et cristallographie
COHEN Joseph	Electrotechnique
DURAND Françis	Metallurgie
MOREAU René	Mécanique
POLOUJADOFF Michel	ELectrotechnique
VEILLON Gérard	Informatique fondamentale et appliquée
ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM. BOUVARD Maurice	Génie mécanique
CHARTIER Germain	Electronique
FOULARD Claude	Automatique
GUYOT Pierre	Chimie minérale
JOUBERT Jean Claude	Physique du solide
LACOUME Jean Louis	Géophisique
LANCIA Roland	Physique atomique
LESPINARD Georges	Mécanique
MORET Roger	Electrotechnique nucléaire
ROBERT François	Analyse numérique
SABONNADIERE Jean Claude	Informatique fondamentale et appliquée
Mme SAUCIER Gabrièle	Informatique fondamentale et appliquée

MAITRE DE CONFERENCES ASSOCIE

M. LANDAU Ioan Doré	Automatique

CHARGE DE FONCTIONS DE MAITRES DE CONFERENCES

M. ANCEAU François	Mathématiques appliquées

Fait à Saint Martin d'Hères JANVIER 1974

Je tiens à exprimer ici toute ma reconnaissance à Monsieur le Professeur J. KUNTZMANN, qui a bien voulu me faire l'honneur de présider le jury de cette thèse,

et aux autres membres du jury

Monsieur le Professeur M. GRIFFITHS, qui m'a aidé lors de mes débuts en Informatique et Messieurs H. GALLAIRE et J. C. SABONNADIERE

qui m'ont tous permis d'améliorer la forme et le fond de ce travail par leurs nombreux conseils

Monsieur J. MERMET, le responsable de l'équipe où ce travail a été effectué, qui a bien voulu m'accueillir et me donner assez d'autonomie pour que ce travail puisse être commencé, puis amélioré.

Rien n'aurait pu être mené a bien sans le concours de nombreux participants qui n'ont pas ménagé leurs peines ; qu'ils ne croient pas que je les ai oubliés parce que je ne les nomme pas : je crains d'en oublier, je les en remercie sincérement. Toutefois, je tiens à préciser que les propos qui suivent ne peuvent engager que moi, l'exploitation de tout ce qui a pu être dit ou fait m'étant personnelle.

Je remercie également le service tirage pour sa rapidité et pour le soin qu'il a apporté à la réalisation matérielle de cet ouvrage.

INTRODUCTION

CHAPITRE I

METHODE D'APPROCHE

CHAPITRE II

SYSTEME SIGMA-C.A.O.

CHAPITRE III

APPLICATION: CASSANDRE

CHAPITRE IV

APPLICATION: ARCHITECTURE

INTRODUCTION

L'ensemble de l'étude que nous présentons porte sur des aspects de la conception assistée par ordinateur (C.A.O.).

0.1 Qu'est-ce-que la C.A.O. ?

Il nous semble que la meilleure façon de rendre compte de ce qu'est la C.A.O. est de citer un certain nombre de définitions en commençant par celle de D.T. ROSS(15), définition utilisée en 1972 lors de la conférence "IFIP working conference on principles of Computer-Aided Design" (20):

la C.A.O. est une technique dans laquelle l'homme et la machine constituent un tandem (pour la résolution de problèmes) en couplant intimement les meilleures caractéristiques de chacun pour que ce tandem travaille mieux que chacun seul, pour offrir la possibilité de travail intégré d'équipe en utilisant une approche pluridisciplinaire.

On peut se rendre compte de l'évolution qu'il y a eu dans la définition de la C.A.O. en regardant (2), (16), (17). C'est la collaboration entre les participants qui a été surtout soulignée en évoquant leur complémentarité souhaitable (et bien souvent évidente) dans le processus de conception; citons (17): "l'ordinateur a un rôle unique à jouer dans le processus de conception, supplémentaire à celui

du concepteur. Les tâches de conception sont aujourd'hui si complexes, et contiennent tellement de contraintes techniques et économiques, que l'introduction de l'ordinateur, avec ses capacités propres, représente la seule voie permettant d'obtenir un "objet" faisable".

Nous voyons la supériorité de l'homme dans son intelligence, sa possibilité de réflexion (avec rapidité) et de prise de décision; par contre la machine dispose d'une capacité de stockage, de la puissance et de la vitesse de calcul, ainsi que d'une grande fiabilité.

Précisons encore en citant (13): "l'homme a tendance à résoudre les problèmes d'une façon heuristique, tandis que l'ordinateur les résoud à l'aide d'algorithmes; c'est-à-dire que l'homme arrive à la solution pratique par essais avec erreurs, tandis que l'ordinateur obtient la solution précise en suivant une séquence de traitements logiques, sans erreur. Ces deux types de solution sont utiles dans beaucoup de problèmes de conception; en permettant à l'homme et à l'ordinateur de travailler efficacement dans la partie du problème pour laquelle il est le mieux armé, on obtient un meilleur résultat que par un effort individuel de chacun".

L'importance que l'on donnait surtout à la collaboration entre l'homme et la machine est maintenant portée à la notion d'intégration au sein d'un même système de différentes phases de la conception (9):

la conception assistée par ordinateur est concernée par la création des données qui décrivent l'objet à concevoir, la

manipulation de ces données afin d'aboutir à une forme achevée de conception, et la génération des informations nécessaires à la fabrication de cet objet à partir des données de description de la conception stockées dans l'ordinateur.

Du point de vue informatique le domaine de la C.A.O. est à la charnière de l'informatique appliquée et des domaines d'application; quiconque y travaille doit aborder d'une façon occasionnelle de nombreux domaines (langage, système, algorithmique, ...) sans pour autant espérer obtenir des résultats remarquables dans chacun d'eux. Finalement ce qui compte dans la C.A.O. c'est le fonctionnement de l'ensemble: le but est de constituer un système valable, équilibré, facile d'emploi et peu coûteux, tout en étant le plus proche de l'état d'avancement des recherches dans les différents domaines et, à la limite, en influençant des axes de recherche.

0.2 Processus de conception

Avant d'aborder plus en détail la C.A.O. essayons de préciser ce que nous entendons par processus de conception.

La figure 0-1 schématise les différentes phases de la vie d'un produit avec les indications sur les influences mutuelles.

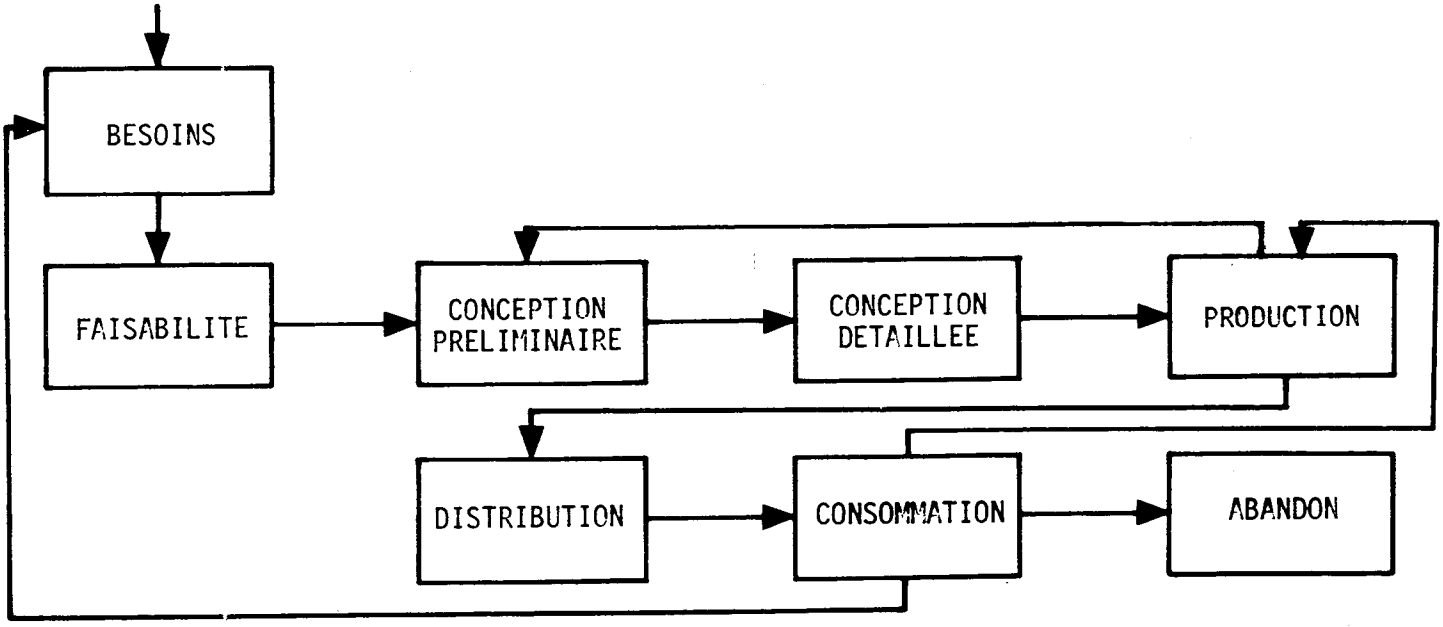


Fig. 0 - 1 Différentes phases de la vie d'un produit

-- la première consiste en l'analyse et la formulation du ou des besoins sur le produit.

-- la deuxième vérifie la faisabilité du produit, choisit une méthode de réalisation et précise la nature des paramètres.

-- la troisième consiste en une conception préliminaire (ou grossière) ; c'est la résolution par la méthode délivrée par la phase 2. La solution obtenue est testée et validée, souvent en modélisant et simulant.

-- la quatrième consiste en la conception détaillée, en tenant compte des possibilités et intérêts de production.

-- les phases suivantes production, distribution, consommation et abandon du produit ne nous intéressent pas particulièrement.

Le processus de conception proprement dit doit tenir compte de la remarque fondamentale suivante:

Comme la conception d'un objet est un compromis entre plusieurs caractéristiques souvent contradictoires, et comme les critères de test dépendent de ces caractéristiques, en absolu, on ne peut jamais être certain d'obtenir le meilleur objet (solution), mais seulement par rapport à une certaine hiérarchie de contraintes.

Ce processus est schématisé sur la fig. 0-2.

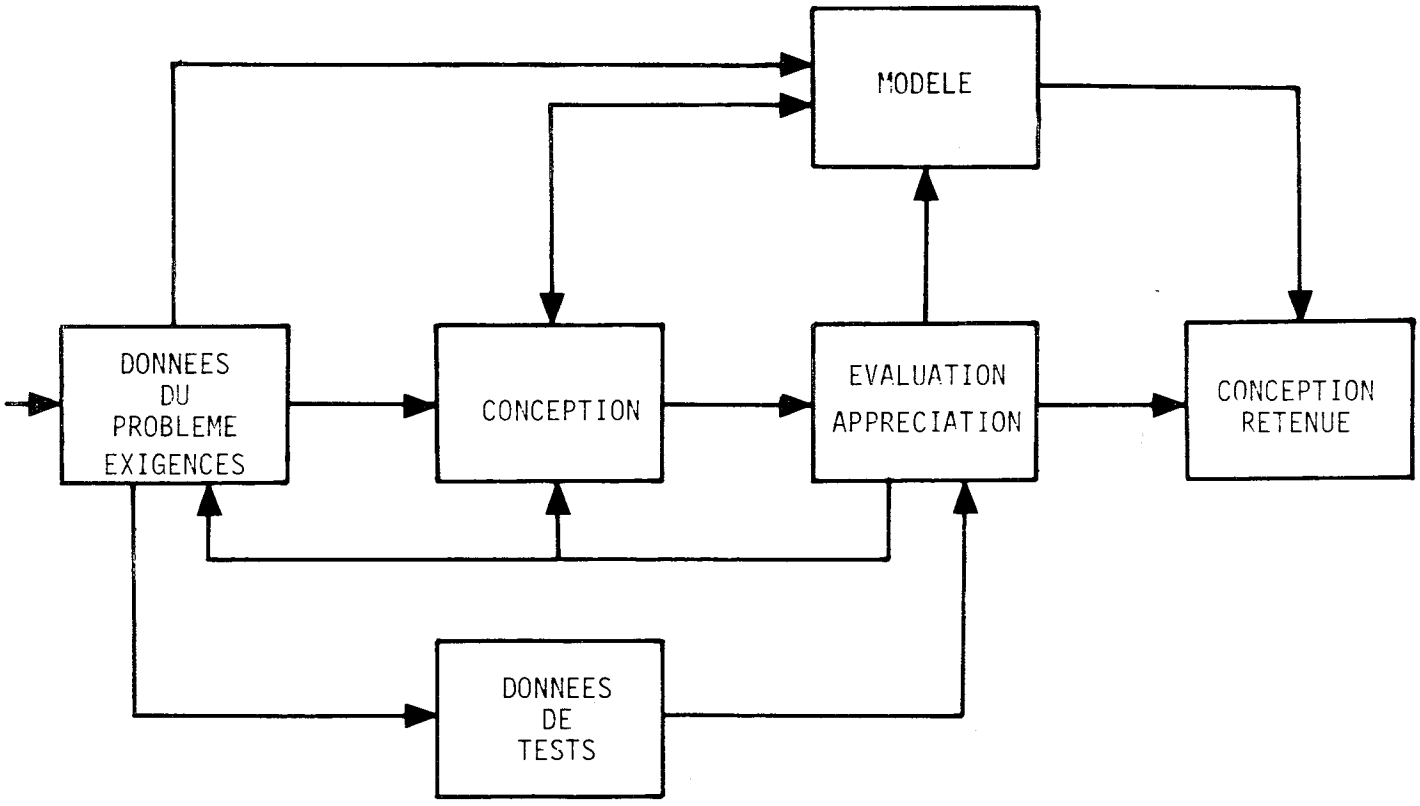


Fig. 0 - 2 Processus de conception

C'est un processus itératif: en partant de la description des données et des exigences on procède à la conception d'un objet que

l'on analyse et teste, pour pouvoir porter un jugement, d'un point de vue quantitatif et qualitatif; éventuellement on modifie les données et les exigences que l'on réintroduit dans le processus de conception.

Lors de différentes itérations on prend en compte des données de plus en plus précises et variées, ce qui peut être vu comme une séparation en différentes phases de conception (voir figure 0-1 une séparation pré-conception, conception fine).

Remarque: Il s'agit évidemment d'une abstraction qui a été faite car on espère retrouver ce processus dans la plupart des cas.

0.3 Historique

Utilisée depuis longtemps dans le domaine de l'électronique la C.A.O. a été récemment adoptée comme outil de résolution dans d'autres domaines (très variés) et depuis quelques années elle acquiert une importance de plus en plus grande dans le monde de l'informatique. On peut considérer comme moment important pour le développement de la C.A.O. l'intervention de S.A. Coons (2), D.T. Ross(14) et I.E. Sutherland (19) lors de la conférence SJCC en 1963, où il faut remarquer que les principaux concepts de la C.A.O. ont été déjà formulés.

Les grands développements pendant les dix dernières années des systèmes conversationnels et des terminaux graphiques (5), (8), (13) ont largement contribués à l'extension des techniques de la C.A.O..

Du point de vue historique nous pouvons distinguer quatre étapes dans l'évolution de la C.A.O..

Dans une première étape on utilisait l'ordinateur pour exécuter une série uniquement temporelle de programmes, chaque programme étant construit pour travailler d'une façon autonome et accomplir une tâche précise correspondant à tel ou tel aspect du processus de conception dans un domaine déterminé (calcul de circuits par exemple). Souvent il s'agissait de la phase d'évaluation. Le passage d'un programme à un autre s'effectuait d'une façon manuelle et nécessitait donc une transcription manuelle d'informations.

Dans la deuxième étape on trouve la réalisation de programmes intégrés liés toujours au domaine d'application bien précis mais formant un tout et aidant donc, avec des degrés divers d'automatisation, à la conception d'un objet déterminé. Les trois parties essentielles d'un tel programme sont:

-- un ensemble de procédures de manipulations et de calculs
-- une structure de données pour le stockage des informations communes
-- un analyseur du langage d'entrée

On peut se demander à quel point ces différentes parties sont directement liées au domaine d'application. Les fonctions de manipulations et de calculs sont bien présentes quelque soit le domaine d'application mais les programmes sont étroitement liés à une application particulière. Ceci n'est plus vrai pour les deux autres composantes (structure interne de données et langage de description). C'est cette constatation qui forme l'amorce d'une évolution vers une troisième étape de la C.A.O..

Cette troisième étape conduisant aux systèmes généraux pour la C.A.O. peut être caractérisée de deux façons:

-- sur le plan technique il s'agit de rendre les fonctions de stockage et de description indépendantes de l'application et de fournir une méthode standard pour la mise en oeuvre à partir du noyau des programmes d'application.

-- sur le plan de la conception il s'agit de rendre effectifs les deux aspects fondamentaux de la C.A.O., que nous avons évoqué dans 0.1, voir l'association homme-machine et une véritable intégration ou l'automatisation de la conception.

Néanmoins l'utilisation d'un tel système pose un certain nombre de problèmes dont les deux principaux semblent être:

-- efficacité

-- facilité de manipulation

En effet des outils généraux (prévus pour une utilisation générale) peuvent paraître inefficaces dans le contexte bien déterminé d'une application précise et, parce qu'ils offrent des possibilités multiples, ils seront peut être difficilement manipulés par un utilisateur non averti. En plus, liés à une structure matériel-logiciel figée, ils peuvent difficilement suivre l'évolution vers les systèmes distribués.

Ces deux raisons fondamentales nous conduisent à entrevoir une quatrième étape d'évolution de la C.A.O. caractérisée par le fait que l'on recherche l'efficacité du système C.A.O. aussi bien pour son utilisation (facilité de manipulation) que dans sa conception (efficacité et facilité de manipulation).

Ceci fera apparaître une distinction entre le concepteur de système C.A.O. et ses utilisateurs. C'est le concepteur qui disposera d'un système général et d'outils complexes de définition et de génération de systèmes particuliers, simples et efficaces, utilisables facilement par les utilisateurs non avertis venant du domaine d'application en question (et donc pas nécessairement informaticiens).

C'est ce principe qui constitue le fondement du système SIGMA - CAO et nous allons poursuivre l'explication en présentant la démarche qui nous y a conduit.

0.3 Pour une généralisation des systèmes C.A.O.

0.4 Notre démarche

Nos travaux, qui ont débuté il y a trois ans, sont effectués dans l'équipe "Méthodologie de la C.A.O." de l'ENSIMAG. Nous pouvons distinguer trois phases dans leur déroulement:

Nous avons commencé par le développement d'un programme d'édition et de manipulations de schémas logiques pour le système CASSANDRE (1). Dans un deuxième temps nous nous sommes intéressés à l'utilisation de la C.A.O. dans la conception architecturale, domaine nouveau et vaste aux multiples possibilités et problèmes. Cette étude commençait par une phase bibliographique puis s'est poursuivie par le développement d'outils ponctuels pour la résolution de problèmes particuliers. Lorsque nous nous sommes posés le problème d'intégration de ces outils au sein d'un système unique nous avons constaté certaines similitudes entre les deux applications précédentes et d'autres travaux de l'équipe (3), (11). Ceci nous a conduit à étudier dans un troisième temps l'implémentation de systèmes C.A.O. d'une façon plus systématique.

Nous nous sommes fixés comme objectif de concevoir un système-générateur de systèmes particuliers et indépendant du support informatique sous-jacent et cette étude a abouti à la définition du système SIGMA-CAO (système interactif graphique pour de multiples aides dans la C.A.O.) dont les principales caractéristiques sont les suivantes:

Le système S.I.G.M.A.-C.A.O.:

-- est basé sur des concepts généraux pour donner la possibilité de traiter des applications variées.

-- introduit un modèle pour la formulation et manipulation d'objets pour permettre un travail naturel et commode.

-- introduit la notion de phase de conception pour permettre une conception intégrée donc une résolution plus globale et mieux contrôlée.

-- dispose d'outils de génération de systèmes spécialisés pour faciliter la tâche de l'implémenteur et de l'utilisateur.

-- introduit la notion de machine abstraite pour permettre une optimisation de fonctionnement d'un système spécialisé et pour assurer une portabilité aussi grande que possible.

Dans ce qui suit nous allons présenter l'approche puis le formalisme servant de support (chapître I). La définition d'une machine abstraite basée sur ce formalisme, sa structure et son fonctionnement, étant abordés dans le chapître II. La présentation d'outils pour l'implémentation d'une application donnée dans ce contexte terminera ce chapître.

Puis nous présenterons dans les chapîtres III et IV les deux applications (systèmes logiques et architecture), qui ont été à l'origine de cette étude et nous montrerons leur comportement dans le nouveau contexte. Etant donné que le système CASSANDRE est suffisamment connu, nous n'en ferons aucun commentaire bibliographique; par contre ceci doit être fait pour l'architecture (début du chapître IV).

Bibliographie

|1| BRESSY Y., DAVID B., FANTINO Y., MERMET J.
A hardware Compiler for Interactive Realisation of Logical Systems described in CASSANDRE
Workshop on Computer Hardware Description Languages and Their Applications / New York / sept. 3-5 / 1975

|2| COONS S. A.
An Outline of the Requirements for a Computer-Aided Design
AFIPS Proc. SJCC / Vol. 23 / pp.299-304 / 1963

|3| DI RUZZA F., LE FAOU C., MERMET J., RECHENMANN F.
Computer Tools for the Construction of Dynamic Space-distributed Simulation Models
Rapport ENSIMAG / 1974

|4| DLUGATCH I.
The applicability of Computer-Aided Design as a System Engineering Tool
Proc. IEEE / Vol.55 / No. 11 / pp.1940-1945 / 1967

|5| Interactive Graphics in Data Processing
IBM Systems Journal / Vol. 7.No. 3-4 / 1968

|6| JACOLIN M., LE FAOU C., PIMORT, VERAN
IMAG2, notice d'utilisation, description du programme, notice de programmation
juillet 1970

|7| JACQUART R., REGNIER P., VALETTE F. R.
GERMINAL: towards a general and integrated system for C.A.D.
Journees sur les systemes generaux pour la C.A.O. / Toulouse / 1974

|8| JOHNSON C. I.
Principles of Interactive Systems
IBM Systems Journal / Vol. 7 / No. 3-4 / pp. 157-173 / 1968

|9| LANG C.A.
Achievements in Computer-aided Design
Proc. IFIP 1974

|10| LATOMBE JC.
Elaboration d'un Systeme Pedagogique d'Assistance a la Conception en Electronique
These de docteur-ingenieur / Grenoble / 1972

|11| LE FAOU C., SICOT J.P.
IMAG III, Systeme de simulation de circuits electroniques
Electronique et microelectronique industrielle / novembre 1974

|12| MILLER P.K., STRATTAN R.D.
Man-Machine Symbiosis: A Conspectus of Interactions
IEEE / vol.IGA-7 / No. 2 / pp. 178-180 / 1971

|13| PRINCE M. D.
Interactive Graphics for Computer-Aided Design
ADDISON-WESLEY Publishing Company / 1971

|14| ROSS D.T., RODRIGUEZ J.E.
Theoretical Fondations for the Computer-aided Design System
AFIPS Proc. SJCC / vol. 23 / pp. 305-322 / 1963

|15| ROSS D.T.
The AED Approach to generalized Computer-aided Design
Proc.ACM National Meeting / pp.367-385 / 1967

|16| Special issue on Computer-Aided Design
Proc. IEEE / Vol. 55 / No. 11 / 1967

|17| Special issue on Computers in Design
Proc. IEEE / Vol. 60 / No. 1 / 1972

|18| SIDERS R.A., et al.
Computer Graphics - A Revolution in Design
Americam Management Association / 1966

|19| SUTHERLAND I. E.
SKETCHPAD: A man-machine Graphical Communication System
AFIPS Proc.SJCC / Vol. 23 / pp. 347-354 / 1963

|20| VLIETSTRA J., WIELINGA R. F. / editeurs /
Computer-Aided Design
Proc. IFIP / Working conf. on Principles of CAD / 1972
North-Holland publishing company / 1973

CHAPITRE I

METHODE D'APPROCHE

I.0 Avertissement

Après énumération des exigences d'un système pour la C.A.O., nous mettrons en évidence le fonctionnement type d'un système conçu pour une application donnée, puis nous aborderons les techniques d'implémentation.

Ces considérations nous amèneront à justifier l'approche que nous avons choisie: c'est pourquoi nous parlerons de formulation de problème, de l'intégration des phases de conception (résolution) et de l'interaction homme-machine.

Nous terminerons en introduisant un formalisme intéressant pouvant servir de support: les Multi-Sur-Reseaux (MSR).

I.1 Position et aspects

I.1.1 Exigences d'un système pour la C.A.O.

Avant d'aborder les systèmes C.A.O. du point de vue du concepteur, essayons de les approcher du point de vue des utilisateurs. Parmi les multiples exigences d'un système pour la C.A.O. les quatre points suivants nous paraissent fondamentaux:

Formulation du problème

C'est un aspect essentiel: en effet pour un traitement informatique il faut obtenir une formulation précise et non ambiguë du problème à traiter.

Pour aborder un problème comportant de très nombreuses informations de type varié, il faut pouvoir l'exprimer successivement avec plus ou moins de raffinement afin de minimiser le nombre d'informations à prendre en compte; ceci nous amène à la hiérarchisation de problèmes; souvent il est utile de pouvoir prendre en compte successivement les différents aspects du problème à traiter, ce qui oblige à manipuler des descriptions incomplètes éventuellement à plusieurs hiérarchies.

Prenons comme exemple une ville: selon que l'on s'intéresse aux écoles, aux espaces verts; à l'électricité, au téléphone, au réseau de transport au réseau piétonier, etc... on n'a pas besoin des mêmes

informations et pour les différents services on ne trouve pas nécessairement les mêmes regroupements; par exemple les zones désservies par une centrale électrique, par une centrale telephonique, par une distribution de courrier ne sont pas nécessairement les mêmes et ne correspondent pas forcément au découpage en quartiers.

Entrée des données

L'entrée des données dans le système doit être commode pour l'utilisateur, claire afin de permettre une vérification rapide; la réalisation en est délicate étant donné que la notion de commodité peut être appréciée différemment selon:

-- la nature de l'information nécessitant éventuellement un périphérique particulier, entrée vocale par exemple

-- les données sont volumineuses ou non: selon le cas une entrée par carte, par bande , par clavier etc..., peut être soit fastidieuse soit commode.

-- la familiarisation avec le système: selon le cas on peut considérer les informations fournies par le système comme assez explicites, suffisantes, trop explicites et même gênantes si elles ralentissent le travail (temps d'impression, volume d'information).

Interaction homme-machine

Pour permettre une intéraction aussi fine que possible les traitements élementaires du système doivent être organises afin de permettre leur enchaînement plus ou moins automatique, selon le désir de l'utilisateur.

Lors de la construction d'un nouveau traitement (algorithme) on apprécie la possibilité d'activer directement les différents traitements mêmes élémentaires, mais lors de l'utilisation de cet algorithme il deviendrait rapidement fastidieux d'activer toujours cette même séquence d'actions, on apprécie donc la possibilite de rendre l'enchaînement automatique.

Intégration

Le travail de conception est un processus iteratif portant successivement sur les différents aspects de l'objet et manipule un ensemble de donnees de plus en plus grand et de plus en plus précis. La disponibilite des données en évolution qui doit être assurée peut l'être par l'utilisation d'une base de données.

I.1.2 Organisation et fonctionnement pour une application

Pour une application donnée on peut envisager le fonctionnement suivant:

L'utilisateur formule son problème à l'aide d'un langage de description. Cette description est traduite dans une forme interne au système et stockée dans une banque de données ce qui permet une disponibilité constante des données.

Le déroulement des traitements est commandé par l'utilisateur à l'aide d'un langage de commande.

Pour rendre agréable et efficace ce travail on utilise au maximum les divers périphériques de l'ordinateur pour présenter les données sous forme appropriée, c'est-à-dire compréhensibles et directement utilisables. C'est pourquoi priorité est souvent donnée aux sorties graphiques, surtout sur un terminal graphique permettant l'interaction.

Exemple: Etude du transport dans une ville

L'utilisateur fournit le plan de la ville et les données caractérisant le trafic (les données de test). Puis il procède à l'activation d'un modèle de fonctionnement du transport et le commande à l'aide d'un langage de commande approprié. A la fin de la simulation

il recupère et dépouille les résultats en utilisant les représentations significatives souvent graphiques: schémas, courbes, histogrammes, etc... Selon les conclusions il peut modifier le plan de la ville, changer les données de test ou choisir un autre modèle de fonctionnement.

I.1.3 Méthodes et techniques d'implémentation

La complexité et la variété croissante des problèmes traités en C.A.O. conduit à une séparation de plus en plus grande entre les concepteurs et les utilisateurs de ces systèmes. En effet tous les deux soucieux de leur efficacité respective tendent à avoir des exigences plutôt contradictoires. Nous constatons que l'utilisateur voudrait faire le maximum de choses sans pour autant sortir de son domaine d'application et devoir donc apprendre des choses nouvelles. Mais le fait d'aborder la C.A.O. à partir d'un problème particulier ou même d'un domaine d'application conduit à la conception d'un ou plusieurs programmes spécifiques, toujours très (ou trop) orientés; on ne peut les réutiliser que très difficilement lors d'un changement, même minime, dans l'énoncé ou la résolution du problème.

Pour orienter efficacement les efforts des concepteurs, il faut considérer la C.A.O. comme une matière en soi, même si elle est très mal placée du point de vue problèmes spécifiques; il faut en dégager les concepts généraux: on doit renverser la situation et ne

s'intéresser à une application particulière qu'en tant que projection de concepts. Ceci parait possible car on constate que l'utilisateur ne s'intéresse pratiquement pas à ce qui se passe à l'intérieur du système pourvu que le temps de réponse soit satisfaisant. Pour une implémentation particulière il faut constituer une equipe pluridisciplinaire contenant des représentants de l'informatique (les concepteurs du système) et des spécialistes du domaine d'application (futurs utilisateurs) et définir et générer un système particulier le plus satisfaisant et le plus efficace possible.

Les outils dont le concepteur de systèmes C.A.O. devrait disposer sont de différents niveaux:

-- Gestion de l'organisation machine (structure du materiel): il faut gérer une configuration à la fois économique et suffisamment puissante, pouvant disposer de périphériques variés afin de rendre son utilisation commode.

On arrive souvent à la configuration suivante (Fig.I-1): un petit calculateur gérant un certain nombre de périphériques (terminaux, terminaux graphiques, table traçante,....) et effectuant le maximum de travaux simples, relié (à distance) à un calculateur puissant (travaillant pour plusieurs petits) à qui on soumet, occasionnellement, des traitements trop complexes, ce dernier pouvant être intégré éventuellement dans un réseau d'ordinateurs(11).

-- Système de base mettant à la disposition des concepteurs les fonctions générales d'un système C.A.O.: base de données et sa gestion, base d'algorithmes et sa gestion, outils standards de communication, outils standards d'enchaînement de travaux, etc.(Fig.I-2).

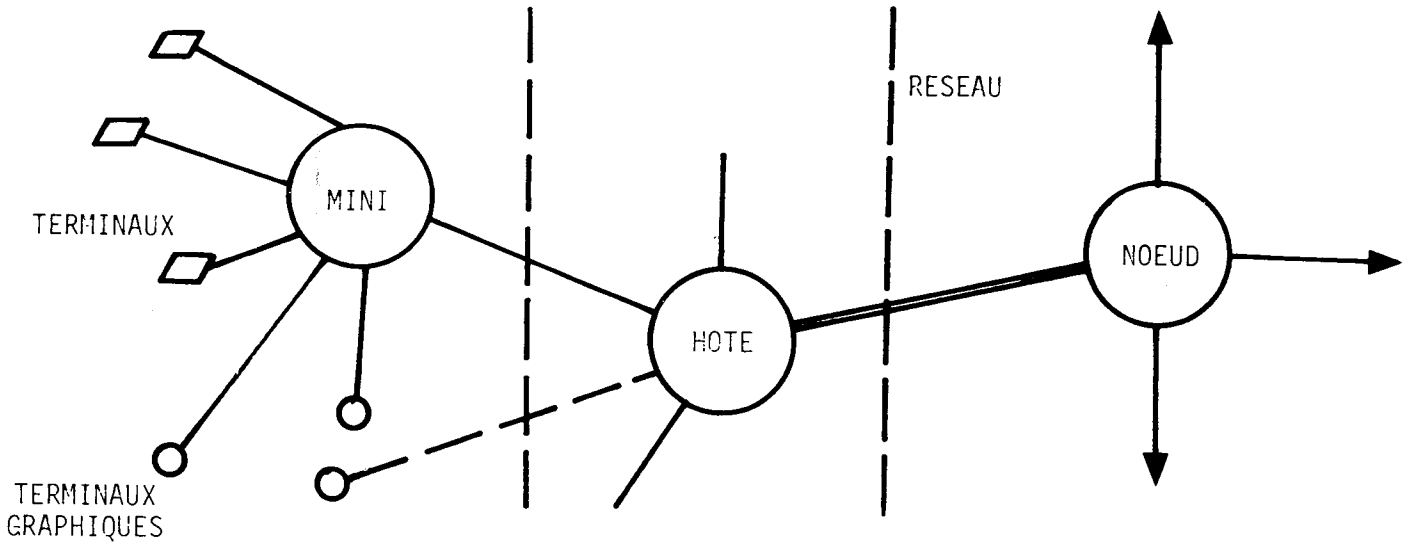


FIG. I-1 Exemple d'une structure de matériel

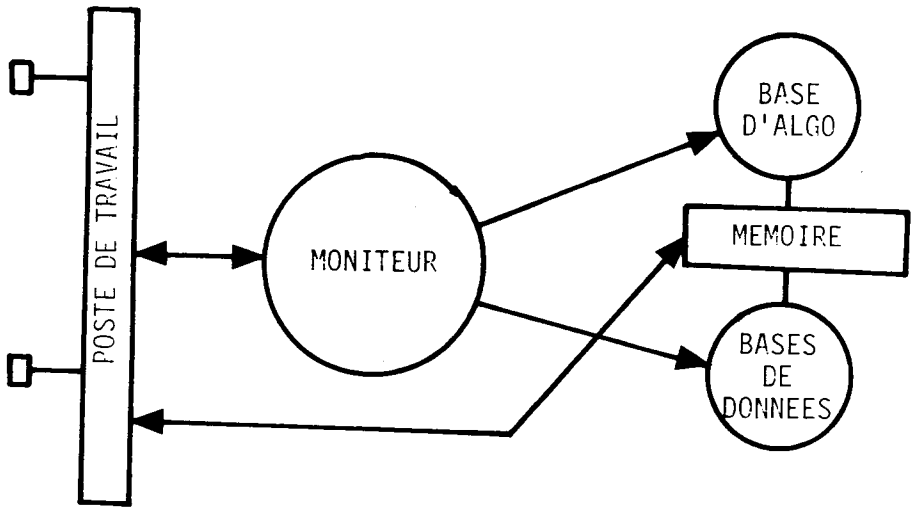


FIG. I-2 Schéma du système de base

-- Logiciel de spécialisation: pour la construction de systèmes spécialisés permettant la spécialisation de la base de données, génération d'un langage de description et de son traducteur, génération d'un langage de commande, etc.; éventuellement une optimisation du système spécialisé. Les outils du troisième niveau doivent se baser sur une représentation cohérente et exhaustive.

I.2 Présentation de l'approche

I.2.1 Justification du choix

Pour concevoir un système ayant les caractéristiques telles que nous les avons esquissées dans le paragraphe précédent, il faut d'abord définir le noyau du système. C'est ce qui nous a amené à introduire un modèle entre les niveaux deux et trois de cette classification. Le modèle proposé consiste en un formalisme avec lequel on peut décrire les applications C.A.O. et en une machine abstraite servant de support pour ce formalisme et faisant le lien avec le deuxième niveau (Fig.I-3). Vis-à-vis des niveaux un et deux, on a ce qui correspond à une implémentation particulière dans un contexte précis sur une structure hardware sous-jacente; vis-à-vis du niveau trois il s'agit d'une définition générale correspondant à la problématique propre de la C.A.O..

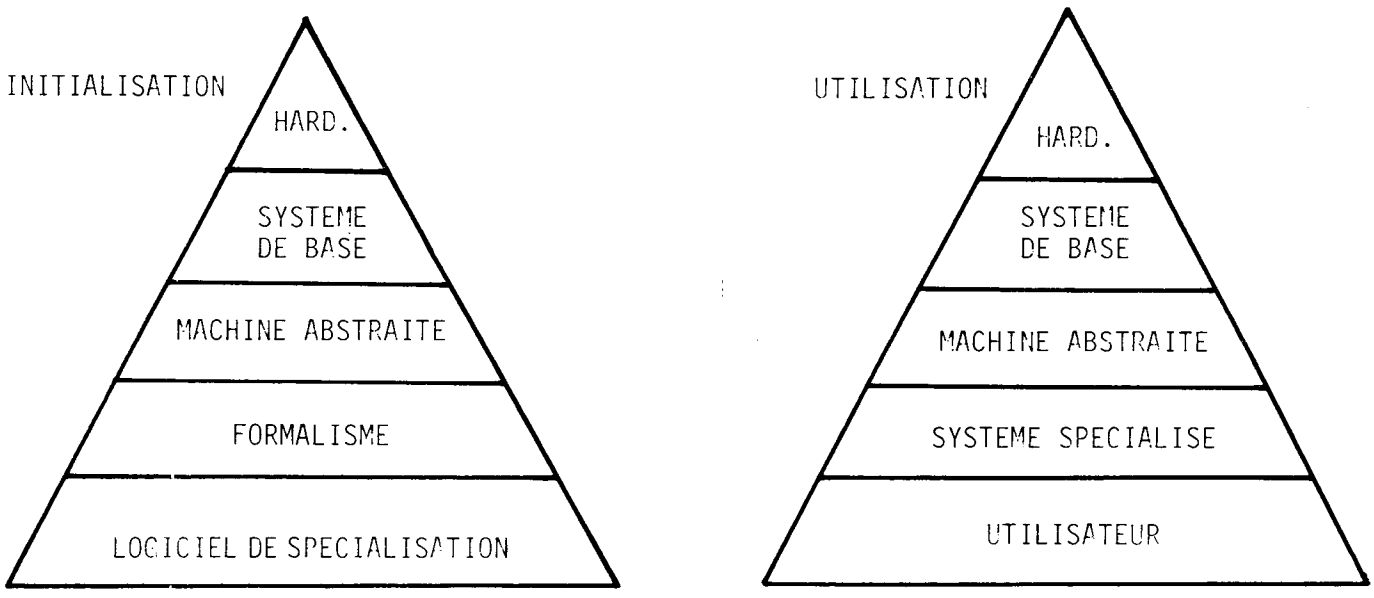


FIG. I-3 Contexte de construction d'un système particulier

Pour définir ce modèle nous avons pris les différentes exigences des utilisateurs et nous avons essayé de dégager ce qui peut être considéré comme commun. Parmi les quatre exigences:

-- Formulation de problème
-- Entrée de données
-- Interaction
-- Intégration

la deuxième (entrée de données) est directement liée à l'implémentation particulière (choix d'un moyen d'entrée approprié) et découle de la formulation de problème (formulation commode pour l'utilisateur) et de l'intégration (utilisation d'une base de données). Nous allons donc passer en revue les trois autres aspects.

I.2.2 Formulation de problème

La première étape dans la résolution d'un problème est sa formulation. On s'aperçoit que si un problème est complexe il est quasiment impossible de le formuler d'un seul coup; de plus, même dans le cas où cela peut être fait, une telle formulation s'avère peu constructive.

Une meilleure approche est de le considérer comme composition d'un certain nombre de sous-problèmes (on devra bien entendu préciser les relations qui les lient).

Le découpage du problème nous paraît fondamental dans le contexte de la C.A.O.. Souvent, une fois le problème bien posé sa solution est immédiate.

La décomposition peut être formalisée sous forme de graphe. Les graphes occupent une place importante en mathématiques appliquées, plusieurs problèmes importants ayant pu être modélisés et résolus ainsi. Il est vrai qu'ils peuvent être remplacés par d'autres techniques, des méthodes booléennes par exemple; mais nous pensons que l'avantage primordial des graphes est de rendre la solution du problème plus visuelle et par là plus intuitive; de plus les graphes facilitent le mode conversationnel. D'autre part par leur aspect général ils peuvent être une charnière entre un certain nombre d'utilisations.

On peut se servir des graphes de deux façons:

-- dans la premiere le graphe est un support de formulation; le problème est de trouver le graphe correspondant au problème à traiter.

-- dans la deuxième le graphe est un support de resolution; on recherche la solution par une méthode travaillant (uniquement) sur le graphe; c'est donc lui l'appui de la methode algorithmique de résolution, généralement non conversationnelle (recherche du chemin critique par exemple).

Il nous semble que ces deux possibilités peuvent être utilisées dans les différentes parties d'un système pour la C.A.O.; la première intervenant surtout au niveau de l'entrée des donnees, la deuxieme au niveau de la resolution (traitement).

Nous pensons qu'il est peu intéressant de ne permettre qu'une seule décomposition, mais que l'on doit généraliser à un nombre quelconque de niveaux de décomposition.

Nous pouvons utiliser, pour formuler un niveau donné, le réseau dont les noeuds correspondent aux sous-problèmes et les connections aux relations. En se plaçant au niveau supérieur le problème considéré précédemment devient un sous-problème, et le réseau est remplacé par un seul noeud. C'est ce qu'on appelle composition hierarchique ou emboîtement hierarchisé de réseaux.

Un noeud de réseau correspond donc à un problème particulier. On appellera propriété une caractéristique propre au noeud et relation une indication de lien entre les noeuds.

La structure interne d'un noeud sera décrite sous forme de propriétés attachées au noeud et éventuellement par le réseau de noeuds le constituant. Avec l'extérieur le noeud communiquera par les relations indiquées qualitativement et quantitativement.

Citons quelques exemples (pris dans différents domaines):

-- un programme informatique peut être considéré comme un emboîtement hiérarchisé de réseaux; les objets en sont les instructions ou les groupements d'instructions, prenons comme exemple une procédure: elle communique avec les autres objets par les relations suivantes: une relation de contrôle (permettant son activation), et les paramètres; le type d'un paramètre est le type de la relation, le mode d'appel (par valeur, par nom ou par résultat) spécifie l'orientation, et la dimension (dans le cas d'une chaîne, d'un tableau, etc...) la quantifie.

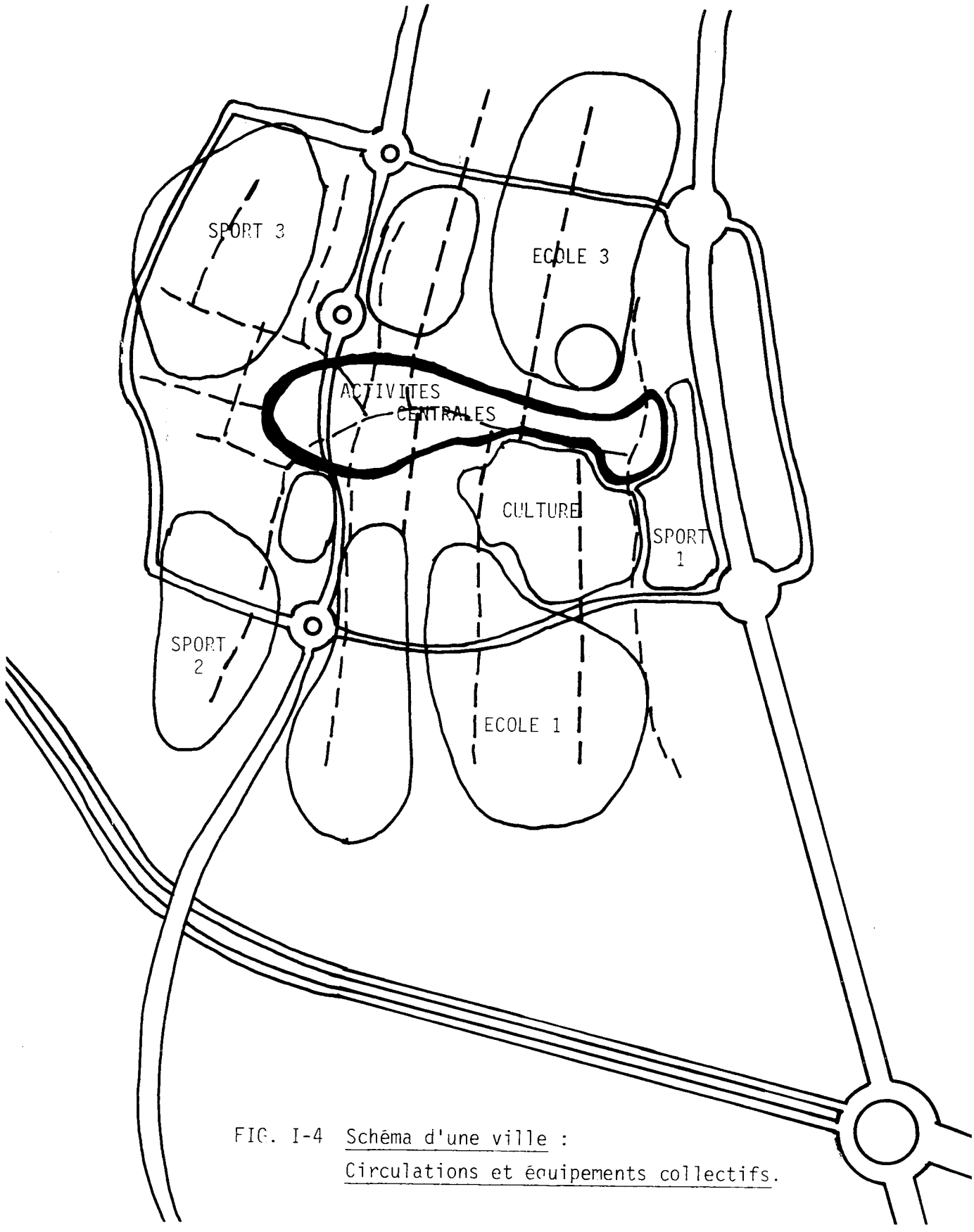


FIG. I-4 Schéma d'une ville :
Circulations et équipements collectifs.

-- une ville est aussi un réseau hiérarchisé avec des zones, des quartiers, des îlots, des maisons comme objets et les circulations, les distributions d'eau, de gaz, etc., comme différents types de relation.

-- une maison contient des appartements, des parties communes pouvant se décomposer en sous ensembles jusqu'aux pieces; avec les differents réseaux: électrique, de gaz, de l'eau, du téléphone, etc..

Le fait que ce type de formulation apparaisse dans de nombreux domaines nous a amené à étudier son implémentation.

I.2.3 Intégration de phases de résolution

Souvent on aborde un problème suivant un certain point de vue, ce qui fait que l'on peut ne s'intéresser qu'aux caractéristiques fondamentales dans le contexte ainsi créé (ce qui diminue le nombre de relations), mais nécessite le traitement de descriptions incomplètes.

Permettre à l'utilisateur de ne s'occuper à un instant donné que des aspects fondamentaux de son problème est un apport considérable qui peut, en particulier, lui servir à une meilleure compréhension lors de la formulation.

Nous avons vu sur la figure 0-1 une séparation pré-conception, conception fine. En réalité nous pouvons obtenir un découpage nettement plus précis; souvent par exemple on distincte trois phases de conception: logique(relationnelle), topologique(géometrique), technique s'effectuant séquentiellement avec possibilité de retour à des phases déjà effectuées (Fig.I-5).

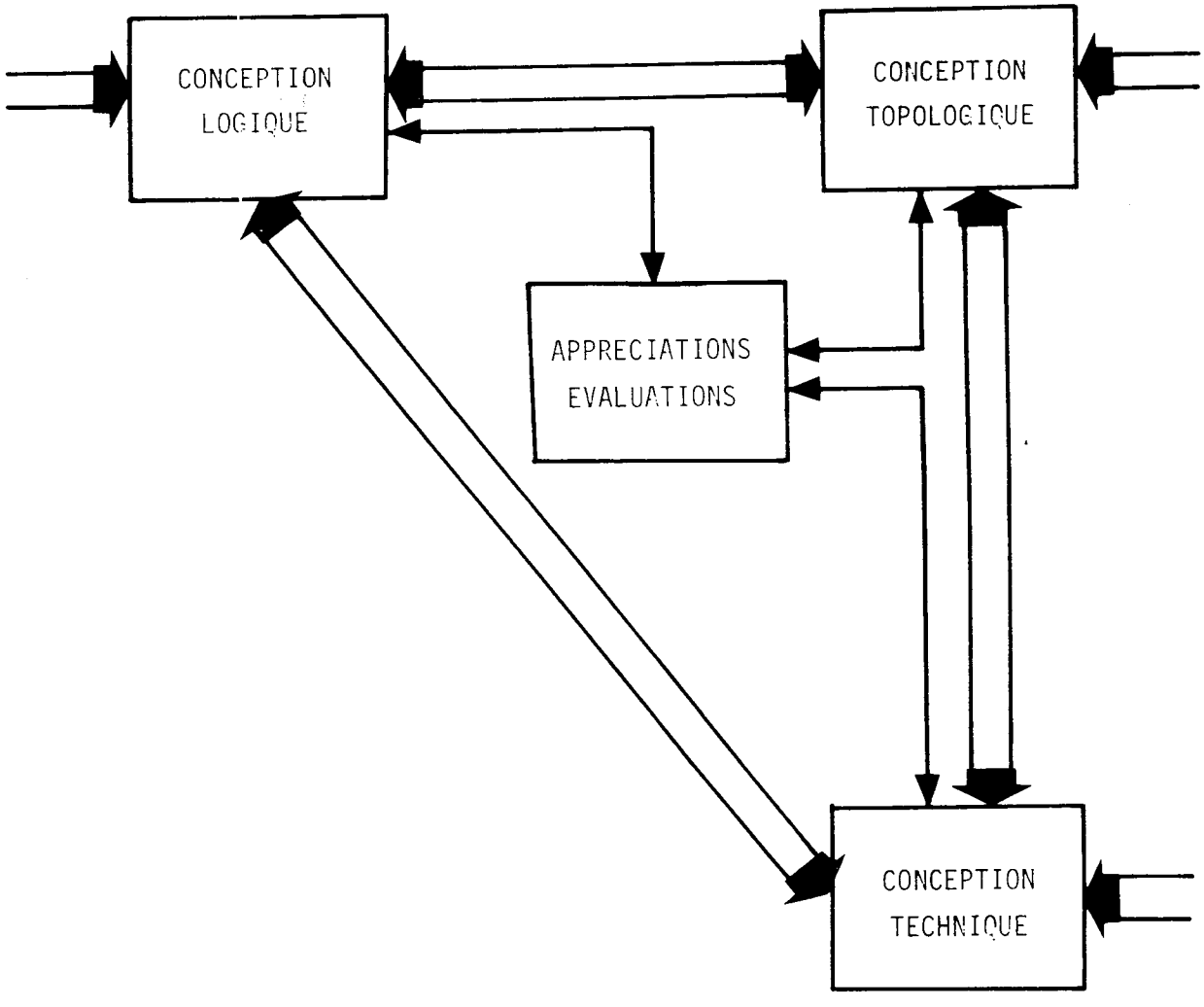


FIG. I-5 Exemple d'une décomposition en phases de conception

-- phase de description logique ou relationnelle où l'on prend en compte le problème et où le traite du point de vue purement logique.

-- phase de description topologique ou géométrique où l'on précise la forme de l'objet et où l'on prend effectivement en compte les indications géométriques.

-- phase de description technique où l'on spécifie les données techniques.

Exemple: conception d'un circuit. Dans la phase logique on décrit sa structure logique et son fonctionnement. Dans la phase technique on choisit la technologie qui va servir à la réalisation. Cette technologie précise d'une part les éléments (boîtiers) à l'aide desquels il faudra réaliser le circuit, d'autre part des contraintes technologiques pour la réalisation (répartition de boîtiers, longueur et croisement des fils, méthode de connection: monocouche, multicouche, etc...), dont certaines induisent des données topologiques.

On a dans l'ordre: phase logique, phase technique, phase topologique.

Remarque:

-- il est évident que la conception de différents objets ne nécessite pas les mêmes étapes; c'est pourquoi il faut

introduire la notion de phase sans figer ni son contenu, ni leur nombre. Ce n'est qu'à la spécialisation du système pour une application donnée qu'on les définit et qu'on spécifie leur enchaînement.

Nous allons maintenant passer en revue ce que l'on doit décrire à chacune des phases, en nous inspirant du découpage précédent en trois phases, en vue de déterminer tout ce qu'il est nécessaire de définir pour la description.

Données logiques ou relationnelles

Elles décrivent les carctéristiques logiques des constituants de l'objet à concevoir et les liens logiques entre ces constituants (exemple: voir chapitre IV description logique d'un centre socio-culturel).

Ces données décrivent tout ce qui est nécessaire pour entreprendre le processus de conception. On y trouve des données textuelles (ayant essentiellement pour but d'informer le concepteur de tel ou tel souhait) car seulement une petite partie des exigences est exprimable numériquement. C'est au concepteur d'interpreter et de faire entrer dans la suite du processus, explicitement ou implicitement, tout ce qui est textuel.

Données topologiques ou géométriques

Elles spécifient la forme de l'objet. Des données comme la surface, le volume, la distance, le lien avec un autre objet, soit interviennent à la définition de forme, soit en découlent.

Dans beaucoup de domaines d'application on peut limiter la géométrie à la description de polygones et à leur composition; on simplifie ainsi les traitements: un contour se représente simplement par le vecteur des côtés. Cette simplification n'est pas toujours possible: par exemple, en mécanique on est tenu de définir une géométrie précise.

Selon la nature de problème on peut préférer décrire la forme de l'objet par la position de ses constituants dans un espace recepteur ou par les positions relatives des constituants (adjacence). Ces deux représentations étant équivalentes on peut évidement passer de l'une à l'autre.

Données techniques

Elles représentent les données techniques propres à chaque application. C'est pourquoi on peut y trouver la description aussi bien d'un matériau (cas, par exemple, de la conception d'un objet en mécanique) que d'une instruction FORTRAN (conception d'un programme FORTRAN par exemple), etc...

Nous pouvons distinguer:

-- matérialisation d'une relation. Exemples: en architecture, la relation de jointivité qui peut être matérialisée par un mur en béton de 10 cm., ou une relation d'accès par une porte à deux battants d'un certain type et de certaines dimensions.

-- matérialisation d'un objet. C'est la spécification technique attachée à un objet tout entier. Elle peut être complexe: on matérialise, à un niveau donné, un objet qui peut ne pas être simple. Exemples: en mécanique: une piece de plomb, en programmation une instruction simple de type affectation. Exemple de matérialisation complexe: en électronique, une plaquette.

Enchaînement de phases

L'enchaînement de phases explicite les liens et les successions possibles entre les phases. Selon la phase de conception dans laquelle on se trouve la signification, l'interprétation et les manipulations des données ne sont pas les mêmes.

La présentation de l'objet dépend de la phase considérée (position dans le graphe d'enchaînement de phases); une présentation sous forme de schéma est souvent utilisée. A partir du moment où la géométrie a été définie, on peut l'interpréter et fournir une

description correspondante. Ceci est également vrai pour les solutions techniques.

Lors de la définition du problème on indique si la donnée doit être prise en considération exclusivement dans une ou plusieurs phases ou pendant toute la durée du processus (portée de la donnée).

Lors de la description du problème on utilise une formulation proche de l'utilisateur donc non directement exhaustive. Pour rendre la description cohérante il faut donc lors de la prise en compte de cette description traiter les inductions.

Le passage d'une phase de la conception à une autre peut se traduire par la transformation de certaines relations; c'est pourquoi on doit permettre la description de règles de transformation.

Exemple: une donnée technique telle que porte entre deux pièces induit, dans la phase logique, une relation d'accès entre ces deux pièces et, dans la phase géométrique, la relation de jointivité entre au moins deux côtés de ces pièces. Une relation d'accès visuel (phase logique) peut devenir dans la phase géométrique une relation sur la distance.

Lors de la résolution du problème il faut pouvoir contrôler la satisfaction de différentes contraintes. Notamment à la fin d'une phase il faut se rendre compte de contraintes non satisfaites et il faut décider comment on va les introduire dans les phases suivantes.

Le même problème de lien entre les données de différents types se pose quand on veut retourner à une phase en amont (déjà traitée), il faut donc abandonner certaines informations obtenues en aval de cette nouvelle phase mais pas toutes pour pouvoir tenir compte de l'expérience acquise.

Exemple: si toutes les contraintes sur la répartition topologique de pièces n'ont pas été satisfaites et par exemple si l'emplacement choisi pour une pièce entraine un dépassement du niveau sonore autorisé; on doit introduire une solution technique effectuant l'isolation sonore.

I.2.4 Traitements

Après avoir abordé la formulation de problème et l'intégration de phases par lesquelles sa résolution doit s'effectuer, nous essayons de dégager les traitements dont le système doit disposer pour permettre la progression du processus de conception.

Comme dans la conception c'est l'utilisateur qui dirige le processus il est fondamental de lui donner la possibilité d'intervention. C'est pourquoi on introduit la notion d'INTERACTION devant permettre à l'utilisateur d'intervenir dans la résolution et de lui demander d'effectuer un certain travail plus ou moins complexe.

Ce travail nous l'appellerons l'ALGORITHME et son but est de traiter un aspect du problème et faire avancer sa résolution. Remarque: une interaction déclanche donc un algorithme et un algorithme peut contenir des intéractions.

Le choix d'interventions et de traitements associés est fondamental pour permettre à l'utilisateur un travail efficace. Notemment au début du processus il s'agit surtout de la formulation du problème: il faut donner des outils de formulation interactive et d'amélioration de la description initiale. Par la suite, dans la résolution du probleme, les opérations de manipulation doivent pouvoir être groupées pour former les traitements suffisament puissants et efficaces. Dans chaque phase de résolution ces traitements peuvent être très differents et leur degré d'automatisation peut varier selon le désir de l'utilisateur (groupement d'actions).

Pour réaliser ceci il faut fournir les opérations de base: les opérations significatives pour l'utilisateur mais d'un niveau suffisament bas lui permettant un travail quasi manuel sur les données.

Ces opérations de base peuvent être utilisées soit dans les intéractions (donc directement par utilisateur) soit dans les algorithmes (donc par programme).

Le système étant basé sur les opérations élementaires (acces aux informations, manipulations de ces informations, liens entre les deux

opérations précedentes), il faut pour une application donnée construire un ensemble cohérant d'opérations de base.

Indépendamment de la sémantique propre à chaque application nous pouvons distinguer cinq classes d'opérations de base:

-- opérations structurelles: changent la structure de la description du problème.

-- opérations d'interrogation: fournissent des indications sur la description sans la modifier.

-- opérations de modification: travaillent sur la description en effectuant des modifications de renseignements (proprietés, relations) et de leurs valeurs.

-- opérations de construction de commandes: permettant la construction de groupement d'opération.

--opérations de contrôle de processus: commandent le changement de phases et activent les transformations associées.

I.3 Multi-sur-réseaux (MSR)

Nous allons maintenant aborder plus précisement les notions que nous venons d'introduire. Nous donnerons des définitions; puis nous expliciterons des opérations travaillant sur la structure choisie.

I.3.1 Définitions

Le point de départ est la notion de RESEAU (graphe)(14). En faisant abstraction de la sémantique, nous pouvons, pour formuler un niveau donné, décrire le réseau dans lequel les noeuds correspondent aux sous-problèmes et les connections aux relations les liant.

Le réseau représente donc les liens entre les sous-problèmes du point de vue ASSOCIATIF.

Chaque sous-problème peut de nouveau être décomposé en plusieurs sous-problèmes qui constituent eux aussi un réseau (représenté au niveau supérieur par un seul noeud). Nous appelerons SUR-RESEAU (SR) cet emboîtement hiérarchisé de réseaux.

UN SUR-RESEAU EST UN RESEAU DANS LEQUEL UN NOEUD PEUT LUI-MEME ETRE UN RESEAU.

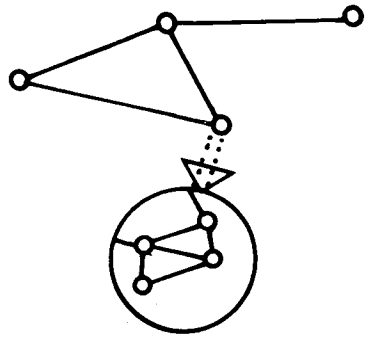


FIG. I-6 Exemple d'un sur-Réseau

Le sur-réseau représente donc les liens entre les sous-problèmes du point de vue associatif et HIERARCHIQUE.

L'opération réunissant les réseaux en un SR est appelée COMPOSITION VERTICALE.

Pour pouvoir introduire la sémantique, des renseignements sont attachés aux noeuds et aux connections.

Les renseignements attachés aux noeuds sont appelés PROPRIETES; le noeud et ses propriétés, considérés ensembles, constituent une ENTITE.

L'information attachée à une connection est appelée RELATION-SEMANTIQUE. Le terme RELATION sera utilisé pour désigner la connection ou le renseignement porté car aucune ambiguïté n'est possible.

Les renseignements présentent deux aspects:

-- qualitatif

-- quantitatif

Etant donné que les réseaux sont généralement considérés comme porteur d'un seul type de renseignement (mono-sémantique), nous définirons un MULTI-RESEAU (MR) comme le réseau multi-sémantique obtenu en considérant des renseignements de différents types.

UN MULTI-RESEAU EST UN RESEAU CONTENANT DES RENSEIGNEMENTS DE PLUSIEURS TYPES.

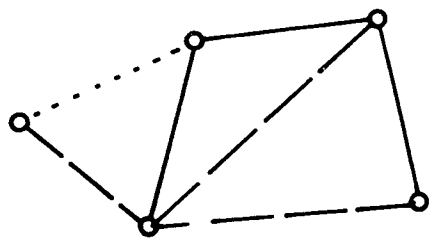


FIG. I-7 Exemple d'un Multi-Réseau

Le sous-réseau d'un MR obtenu en ne considérant des renseignements que d'un seul type sera appelé COUCHE (STRATE).

C'est la composition de ces deux notions qui est à la base de notre formalisme; nous parlerons donc de MULTI-SUR-RESEAU (MSR).

UN MULTI-SUR-RESEAU EST UN RESEAU CONTENANT DES RENSEIGNEMENTS DE DIFFERENTS TYPES DANS LEQUEL LES NOEUDS PEUVENT ETRE DES MSR.

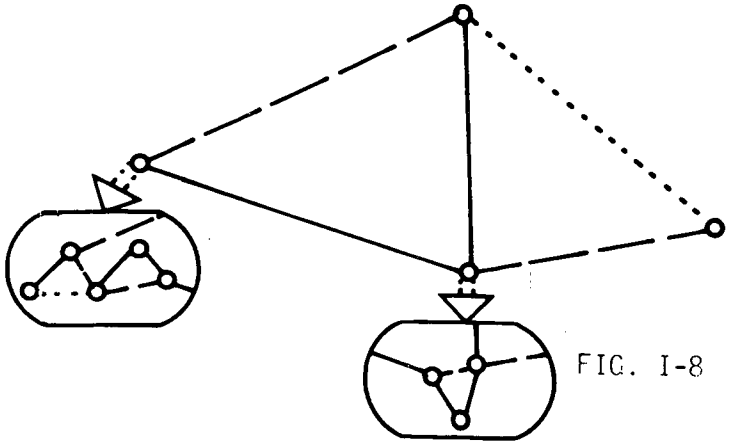


FIG. I-8 Exemple d'un Multi-Sur-Réseau

La structure étant définie, précisons les renseignements qui s'y attachent.

Une ENTITE (noeud et ses propriétés) pour être accessible par un nom qui lui servira de sélecteur. On parlera donc du nom d'un noeud. A chaque noeud on associe deux listes: une liste de propriétés et une liste de liaisons avec les relations.

Un renseignement (propriété ou relation) est défini par les données suivantes:

-- nom
-- type
-- information
-- classe
-- transformation

En plus pour les propriétés on ajoute la notion d'attribut et pour les relations leur orientation.

Explicitons la signification de ces données:

Le NOM sert de sélecteur; il peut être composé indicé mais est facultatif

Le TYPE (nom générique) caractérise globalement un renseignement.

L'INFORMATION spécifie la nature d'information portée par le renseignement et sert pour l'implémentation (entier, réel, tableau,..)

La CLASSE spécifie la portée du renseignement: la phase (ou les phases) lors de laquelle il est actif ou accessible.

La TRANSFORMATION fournit les règles pour la progression du renseignement lors du passage d'une phase à l'autre autrement dit le traitement associé à une donnée lors de la transition dans le graphe d'enchaînement de phases (voir plus loin).

L'ORIENTATION indique si la relation est orientée et dans quel sens.

L'ATTRIBUT a pour rôle de définir le comportement de la propriété dans la hiérarchie. Il est de type hérité ou synthétisé (13) et sa fonction est additive, multiplicative, globale ou particulière.

Un attribut Global transmet la propriété à toutes les entités du sous-MSR. Il est donc du type hérité et empêche la multiplication d'informations identiques et permet une mise à jour rapide. Pour calculer la surface des unités nous pouvons utiliser un attribut du type synthétisé avec la fonction additive si la propriété de type surface se transmet par addition. Le calcul s'effectue d'une façon spontanée et assure donc une mise à jour automatique. Un attribut particulier permet à l'utilisateur de contrôler le comportement du renseignement en spécifiant à chaque fois l'opération qui doit être appliquée ce qui peut être très intéressant dans un environnement conversationnel.

Pour intégrer les différentes phases de conception on définit une PHASE comme une étape de résolution pendant laquelle on n'a accès qu'aux données lui étant associées. La notion d' ENCHAINEMENT de phases est introduite pour définir la succession de phases. Elle est matérialisée par un graphe représentant les phases et les transitions autorisées.

Le passage d'une phase à une autre peut entrainer la transformation des données pour avoir dans la nouvelle phase toutes les informations nécessaires sur l'état d'avancement du processus de conception (voir I.2.3).

Remarque: Une limitation importante pour le formalisme MSR est qu'il ne décrit les problèmes que d'une façon statique. La dynamique se limite à la formulation mathématique (propriétés particulières associées au noeud contenant des équations). Pour prendre en compte une dynamique non numérique deux possibilités sont envisagées:

-- soit on ne la précise qu'au niveau le plus bas ce qui clarifie le programme: on ne mélange pas les concepts.

-- soit on la décrit dans chaque entité: pour chaque problème on peut procéder d'une façon globale à la formulation.

Nous verrons dans le chapître III comment nous avons mis en oeuvre la première éventualité.

I.3.2 Opérations sur les MSR

Les opérations sur le MSR ont pour but de faire progresser le processus de conception. Elles travaillent sur des informations du MSR et, au fur et à mesure de la résolution, les modifient en faisant disparaître certains renseignements, en en introduisant d'autres et en jouant sur les valeurs. Le MSR est donc l'appui de la méthode de conception et il traduit à chaque instant son état d'avancement.

Les opérations élémentaires doivent donc permettre l'accès (création, suppression, modification) à toutes les informations d'un MSR.

Pour un travail de conception ces opérations sont d'un niveau

trop élémentaire; il faut donc fournir les opérations plus puissantes et plus orientées vers la résolution de problème. Ces opérations, que nous appellerons opérations de base, sont construites à partir des opérations élémentaires et peuvent être très différentes selon le problème à traiter. Néanmoins nous considérons que les cinq classes d'opérations sont toujours présentes(voir I.2.4). Passons les en revue et apportons quelques précisions sur leur contenu.

Nous distinguons deux types de STRUCTURATIONS:

-- opérations VERTICALES: changent la profondeur des noeuds dans la hiérarchie des HR du HSR.

-- opérations HORIZONTALES: changent des noeuds sur un seul niveau (sans changer la profondeur).

Il y a deux opérations verticales fondamentales:

-- opération de CONTRACTION: regroupement de plusieurs entités d'un sous-HSR dans une seule de même niveau. Un nouveau niveau est ajouté, formé par les entités contractées.

Remarque: dans l'hypothèse où le HSR contracté se réduit à une seule entité nous considérerons le résultat comme isomorphe au HSR de départ. Il en résulte que le nombre de HSR différents pouvant être obtenus par contraction à partir d'un HSR donné est fini.

-- opération d'EXPANSION: c'est l'inverse de l'opération précédente. Une entité est remplacée par le MSR qu'elle contient.

On peut ramener à deux niveaux tout MSR par expansion de toutes ses entités non terminales; nous nommerons MULTI-RESEAU DE BASE (MRB) d'un MSR le multi-réseau ainsi obtenu.

Remarque: il y a un nombre fini de MSR ayant le même MRB; ils correspondent tous au même problème et ont tous le même fonctionnement (dans le cas où l'on interprète les entités primitives et les propriétés des entités et des connections). La réciproque est fausse: deux multi-réseaux de base différents peuvent parfaitement correspondre aux mêmes problèmes.

Les opérations HORIZONTALES comprennent en particulier les opérations d'union, d'intersection, de complément et de disjonction des entités d'un sous-MSR. Elles interviennent car aucune méthode de formulation ne peut être imposée. Si nous procédons à la formulation d'une façon ou strictement descendante ou strictement ascendante nous obtenons une hiérarchie de sous-problèmes. A partir du moment où nous autorisons le mélange des deux méthodes précédentes et éventuellement faisons appel aux sous-problèmes déjà formalisés, nous pouvons rencontrer des difficultés lors du passage d'un niveau à un autre (partie définie deux fois, ...); le même genre de difficultés se

rencontre si on veut reformuler ou optimiser un problème déjà traité. Des opérations sur les graphes telles que l'union, l'intersection, la différence, etc... peuvent servir d'outil.

Lors de la définition d'opérations horizontales leur portée doit être spécifiée: ou bien elles sont limitées à un seul niveau et travaillent avec les sous-MSR respectifs, ou bien elles sont définies pour tous les sous-niveaux, auquel cas le MRB du sous-MSR doit être utilisé.

La portée des opérations est liée à la définition de l'IDENTITE DE DEUX MSR. Si on la définit comme identité de sélecteurs d'entité les opérations horizontales peuvent être limitées à un seul niveau. Si l'identité est définie comme équivalence des MRB il faut les définir sur tous les sous-niveaux. Dans la première approche c'est la structure hiérarchique qui est importante, dans la deuxième c'est l'ensemble des entités de base et leurs relations.

Les opérations d'INTERROGATION n'effectuent aucun changement sur le MSR. Elles servent à donner des indications, plus ou moins complexes, sur les renseignements contenus dans le MSR. Cela peut aller d'une simple demande sur la présence de telle ou telle entité ou de ses propriétés, jusqu'aux recherches de sous-réseaux d'un certain type (par exemple). Dans le premier cas il s'agit d'activer une opération de base dans le second d'activer un algorithme donc une opération complexe.

Le résultat de telles opérations n'est pas binaire: on obtient effectivement (dans le cas où la réponse est positive) les noms des objets que l'on a cherchés; si on le désire on a ainsi la possibilité d'enchaîner directement, par exemple, une opération de contraction pour isoler ce sous-réseau.

Ces opérations ne sont pas seulement des opérations sur les graphes: pratiquement il est utile d'avoir tous les types d'opérations: par exemple, classification des entités d'un sous-MSR selon un certain critère, calculs à partir des valeurs des propriétés, etc... Nous verrons plus loin des exemples concrets.

Les opérations de MODIFICATION effectuent les changements quantitatifs sur le MSR. Ce sont elles qui assurent l'évolution du MSR dans le processus de conception. De la même façon que pour les opérations d'interrogation elles peuvent être plus ou moins complexes et liées au domaine d'application pour permettre un travail efficace de l'utilisateur.

Les opérations de CONTROLE DE PROCESSUS servent à effectuer les transitions dans le graphe d'enchaînement, effectuent les appels aux transformations pour mettre le MSR dans la forme appropriée et mettent à la disposition de l'utilisateur les opérations liées à la phase courante.

Les opérations de COMPOSITION permettent à l'utilisateur la construction d'opérations complexes à partir des opérations de base et

éventuellement d'autres opérations complexes. Il s'agit d'enchaînement inconditionnel (groupement d'opérations), d'enchaînement conditionnel (instruction si), d'enchaînement répetitif (boucle pour et tantque) avec ou sans passage de paramètres.

Nous avons cherché à pouvoir spécifier des données très variées, d'une façon très simple, et à les faire intervenir, d'une façon naturelle au moment voulu, dans le processus de conception.

En comparant le MSR de départ avec celui d'un certain état d'achèvement du processus de conception, on peut porter facilement un jugement sur la satisfaction des exigences et par là sur l'efficacité de la méthode.

I.4 Exemple: programmation

Avant d'aborder en détail les deux applications (chapîtres III et IV) montrons rapidement l'utilisation de MSR dans un contexte assez manipulatoire, pour l'élaboration de programmes informatiques.

La programmation structurée est actuellement reconnue comme une méthode très importante dans la conception de programmes. Le formalisme MSR peut servir de support pour une conception assistée de ces programmes.

En effet, dans la programmation structurée les problèmes sont analysés d'une façon descendante; à chaque niveau seulement un nombre limité de constructions est utilisé(Fig.I-9).

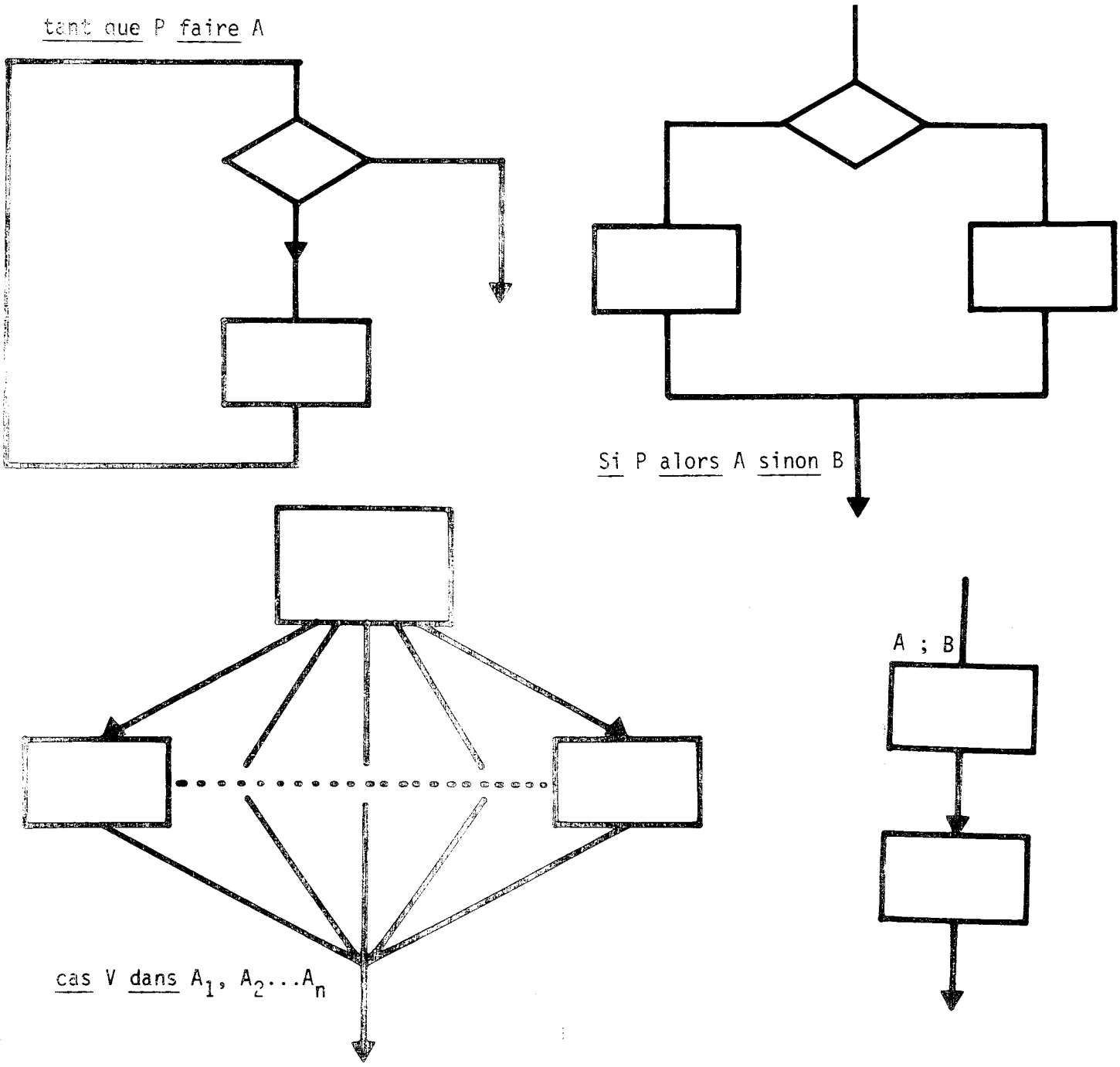


FIG. I-9 Principales compositions de la programmation structurée

La conception d'un programme informatique s'effectue en plusieurs phases. Nous en avons distingué cinq:

-- Définition du problème
-- analyse de processus
-- structuration de données
-- structuration de traitements
-- codage

Dans la phase de définition du problème il s'agit de préciser le problème, c'est-à-dire les données fournies et les résultats escomptés.

Dans la phase d'analyse de processus il s'agit de définir le processus de résolution du problème donné. Pour cela on procède à la décomposition en sous-problèmes et on s'efforce d'obtenir la hiérarchie de ces sous-problèmes.

La structure obtenue est encore assez floue et pour la saisir il suffit de retenir la décomposition verticale et les informations textuelles caractérisant le contenu futur.

Dans la phase de structuration de données il s'agit d'introduire les données, leur structure et de préciser leur place dans la hiérarchie du processus. On fait donc intervenir une autre hiérarchie, la hiérarchie des données.

Les informations sont déjà plus précises et concernent le nom, la structure et la portée (dans la hiérarchie) des données. Pour la saisir on peut définir une structure d'accueil déjà plus précise constituée par les noeuds particuliers "déclaratifs" ayant comme propriétés les déclarations des données.

Dans la phase de structuration des traitements il s'agit de faire correspondre aux informations textuelles les traitements, raffiner la structure hiérarchique, préciser les liens horizontaux et établir les contacts avec la hiérarchie des données pour qu'à la fin de cette phase le processus de conception soit complétement spécifié c'est à dire défini à l'aide des opérations exécutables.

Pour effectuer ce travail la programmation structurée fournit une méthode intéressante en introduisant un nombre limité de constructions (Fig. I-9) à l'aide desquelles on exprime les traitements. Les informations sont donc très précises et pour les saisir on définit une structure d'accueil correspondant à la programmation structurée, c'est à dire les entités particulières correspondant aux contructions de la programmation structurée à l'aide desquelles on va donc construire la version exécutable du processus de résolution. Pendant le travail de construction on peut s'apercevoir de la présence répétée de certaines séquences de traitement. Pour les rendre explicites on a recours aux constructions de procédures, de fonctions et de macro-constructions qui modifient le réseau d'appel et la hiérarchie de données.

Dans la phase de codage il s'agit de traduire le processus décrit avec le formalisme de la programmation structurée dans un langage de programmation (Algol, Fortran; Assembleur, etc...). Ce traitement peut être effectué d'une façon automatique car le nombre de constructions est limité à priori, on peut donc predéfinir leur correspondant dans un langage donné (27).

Remarques: L'intérêt d'une conception assistée nous paraît triple:

-- on fournit un support cohérant permettant une conception raisonnée et guidée, ce qui peut souvent la faciliter.

-- le fait de mémoriser les informations de différentes phases de conception peut résoudre le problème de la documentation de programmes; car ici elle apparaît d'une façon automatique et correspond exactement à la méthode de conception.

-- cette démarche peut aussi apporter une solution au problème de transportabilité et de transcription de programmes. En effet la description du programme dans la quatrième phase est indépendante aussi bien du langage que de la machine, en fournissant les expansions appropriées on obtient le programme pour une machine donnée dans le langage choisi.

Bibliographie

[1] ABRIAL J.R.
Data semantics
Laboratoire d'informatique / USMG / 1973

[2] ALEXANDER Ch.
De la Synthese de la Forme, essai
Dunod / Collection Aspects de l'Urbanisme / 1971

[3] AZEMA J., SAILLARD J. C.
Visualisation interactive de graphes
Rapport / Laboratoire d'Informatique / USMG / 1973

[4] BASKIN H. S.
A Comprehensive Applications Methodology for Symbolic Computer Graphics
in: Pertinent Concepts in Computer Graphics
University of Illinois Press / 1969

[5] BASKIN H. S., MORSE S. P.
A Multilevel Modeling Structure for Interactive Graphic Design
IBM Systems Journal / Vol. 7 / No. 3-4 / pp. 218-228 / 1968

[6] BELADY L. A. et al
A Computer Graphics System for Block Diagram Problems
IBM Systems Journal / Vol. 10 / 1971

[7] BRANIN F. H.
Computer Methods of Network Analysis
Proc. IEEE / Vol. 55 / No. 11 / pp. 1787-1801 / 1967

[8] CROSS n. / Ed. /
Design Participation
ACADEMY Editions 1972

[9] DAVID B., Rivero V.
An Approach to Computer-Aided Design
Proc. Informatica 74 / Octobre 1974

[10] EVANS D. S., KATZENELSON J.
Data Structure and Man-Machine Communication for Network Problems
Proc. IEEE / Vol. 55 / No. 7 / pp. 1135-1144 / 1967

[11] FREEMAN G.C.
Software tools, Hierarchical computer systems and software implications.
Special hardware resources.
Journees sur les systemes generaux pour la CAO / Toulouse / 1974

|12| Journees sur les systemes generaux pour la CAO
Toulouse / decembre / 1974

|13| KNUTH D.E.
Semantics of context-free Languages
Mathematical Systems Theory / Vol.2 / no.2 / pp.127-145 / 1968

|14| KUNTZMANN J.
Theorie des resaux-graphes
Dunod / 1972

|15| LATOMBE JC.
Elaboration d'un Systeme Pedagogique d'Assistance a la Conception en Electronique
These de docteur-ingenieur / Grenoble / 1972

|16| LEFEBVRE J.
Traitement des reseaux sur ordinateur
Rapport / Departement d'Informatique / Universite de Montreal / 1972

|17| LORHO B.
De la definition a la traduction des langages de programmation: methode des attributs semantiques.
These d'etat / Toulouse / 1974

|18| MANHEIM M. L.
Hierarchical Structure: a Model of Design and Planning Processes
M.I.T. Report No. 7 / M.I.T. Press / June 70

|19| MARTIN J.
Design of Man-Computer Dialogues
Prentice-Hall / 1973

|20| MERMET J
Etude methodologique de la conception assistee par ordinateur du systemes logiques: CASSANDRE
These d'etat / USMG / 1973

|21| MOORE G. T.
Emerging Methods in Environmental Design and Planning
The MIT Press / 1970

|22| PAULUS W.K.
Methodological Aspects of Problem Formulation
IEEE / Vol. SSC-2 / No. 1 / pp. 16-22 / 1966

|23| REYNAUD GAROCHE F.
Theorie des ecoulements dans les reseaux
These 3e cycle / USMG / 1973

|24| ROY B.
Algebre moderne et theorie des graphes
Tome 1 et 2 / Dunod / 1969-70

|25| RUSSO R. L., WOLFF P. K.
A Computer-Based-Design Approach to Partitioning and Mapping of Computer Logic Graphs
Proc. IEEE / Vol. 60 / No. 1 / pp. 28-34 / 1972

|26| TURCAT C.
Isomorphisme, immersion et recouvrement de graphes
These 3e cycle / USMG / 1974

|27| VOIRON J., COURTIN J
Une initiation possible a l'informatique
Enseignement de la programmation
Seminaire / Laboratoire d'Informatique / USMG / 1973-74

|28| WEGNER P.
The Vienna Definition Language
ACM Computing Surveys / Vol.4 / no. 1 / pp.6-63 / 1972

CHAPITRE II

SYSTEME SIGMA-C.A.O.

II.0 Avertissement

Ce chapître présente les traits importants du système SIGMA-C.A.O..

Après la définition de la structure de la machine abstraite basée sur le formalisme MSR, nous étudierons son fonctionnement.

Dans le deuxième temps nous présenterons les différents types d'utilisation du système. Nous ferons la distinction entre l'utilisation par le concepteur de système C.A.O. et les deux catégories d'utilisateurs. Puis nous montrerons sur un exemple comment implémenter une application. Ceci nous amènera à dégager les moyens de rationnalisation de ce processus. Il s'agira de la construction des langages spécialisés (langage de description et de commande), d'une méthode de construction du système par maquette de l'application et d'une technique de modèle. Nous présenterons un sous-problème intéressant: le modèle pour les manipulations de schémas.

Nous terminerons par la présentation de trois cas-types de l'implémentation de la machine abstraite sur une machine unique.

II.1 Machine abstraite

II.1.1 Structure de la machine

Pour mettre en oeuvre le formalisme MSR nous avons construit un support informatique sous forme d'une machine abstraite. Ceci nous permet de ne pas melanger dès le début les deux aspects fondamentaux à savoir le fonctionnement intrinsèque du système et son implémentation.

Nous pensons que de cette façon on peut faciliter l'implémentation du système sur des supports matériels différents ce que nous considérons comme très important dans le domaine de la C.A.O. car selon le cas la configuration optimale du matériel peut être soit un miniordinateur soit un réseau d'ordinateurs avec peut-être des bases réparties. De la même façon l'implémentation peut s'effectuer selon les exigences particulières soit sur une machine nue ou même au niveau de la microprogrammation, soit sous un système opératoire à un ou plusieurs niveaux. Les systèmes comme APL, LISP, GERMINAL, BAL, ESPACE, etc... peuvent selon le cas servir de support.

Regardons maintenant l'organisation logique que nous avons retenue (Fig.II-1):

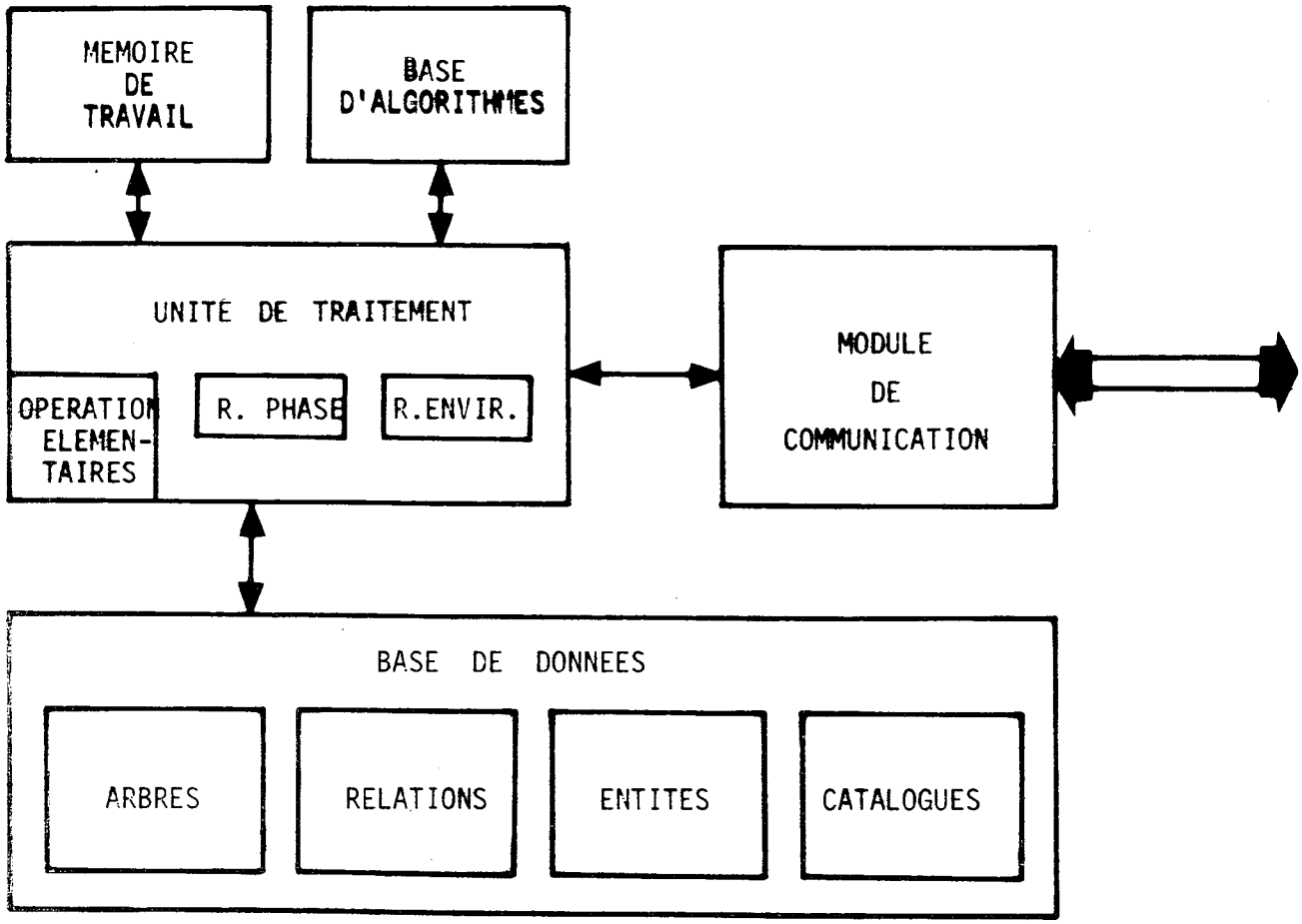


FIG. II-1 Organisation de la machine abstraite

-- une unité de traitement capable d'exécuter les opérations élémentaires et ayant deux registres principaux indiquant respectivement la phase et l'environnement (voir plus loin) dans lesquels la machine se trouve.

-- un module de communication chargé d'effectuer les échanges entre les différentes parties du système et le milieu extérieur. Il devra donc gérer tous les types et modes d'échanges (graphique ou non, données directes ou sous forme de langage, etc...).

-- une base de données assurant la disponibilité des informations contenues dans le système. Elle est interrogée et mise à jour par l'unité de traitement.

-- une banque d'algorithmes contenant pour une application les opérations de base et les opérations plus complexes réalisées à partir des opérations de base à l'aide de langage de commande et éventuellement les traitements (programmes) conçus en dehors du système et présentant un intêrêt certain pour cette application.

-- une mémoire de travail servant pour le stockage temporaire d'informations intermédiaires lors de l'exécution d'algorithmes.

Dans la banque de données nous pouvons distinguer quatre parties:

-- l'aspect hiérarchie (description verticale) est traduit par une structure arborescente. Cette structure rend compte du découpage du problème et intervient dans toutes les opérations structurelles. Remarque: Pour un problème donné on peut avoir au même moment plusieurs hiérarchies traduisant chacune un aspect du problème (voir exemples chapitre I).

-- les entités en présence avec leurs propriétés propres; au départ chacune contient le nom et les caractéristiques directes; lors des traitements ces caractéristiques peuvent être modifiées et enrichies (topologie, solution technique par exemple).

-- l'aspect associatif est représenté par les relations avec leurs caractéristiques et les entités les mettant en jeu.

-- les données partagées se trouvent dans un catalogue. Il s'agît notamment d'entités de base ou des informations intervenant en tant que valeurs dans les renseignements (solutions techniques par exemple). L'utilisation d'une donnée du catalogue dans la description se faisant dans les deux sens on peut à chaque instant aussi bien retrouver tous les endroits de l'apparition d'une donnée du catalogue que connaître toutes les caractéristiques de la donnée utilisée dans la description.

II.1.2 Module de communication

Comme on a déjà pu le constater, dans un contexte de C.A.O. l'aspect conversationnel est fondamental. Dans beaucoup de domaines et en particulier ceux auxquels on s'est intéressé, la communication graphique, même rudimentaire, permet une saisie globale plus rapide que des pages de texte écrit. S.A.COONS (8) évoque l'utilité de se servir d'un dessin qu'il considère comme "un pont naturel entre le

vague incitateur de l'imagination et la description détaillée de concept". Aussi nous sommes nous penchés sur les possibilités d'utilisation des terminaux graphiques dans un tel système. Etant donné la vision globale que permet le terminal graphique, son utilisation dans un environnement conversationnel nous paraît être un outil aux possibilités considérables: l'utilisateur est en liaison constante et étroite avec son système et peut, continuellement, en faire varier les divers paramètres: toute décision, pouvant servir à donner une nouvelle orientation à la résolution du problème, est instantanément transmise au système.

Actuellement les terminaux graphiques disposent des outils de communication bidirectionnelle dont le système peut se contenter: photostyle, manche à balais, clavier alphanumérique, touches de fonctions, etc...

L'aspect graphique n'est pas essentiel en C.A.O.. C'est un support utile certes, unique même pour certains domaines, mais un support seulement: il doit être commode d'emploi et étant donné qu'il s'agit d'un outil fonctionnel il est souhaitable d'étendre son utilisation aux différents domaines d'application.

Les représentations graphiques peuvent, pour les différentes applications, vite diverger. On ne peut espérer concevoir un système graphique global ne faisant pas appel à des programmes spécialisés décrivant la géométrie dans un contexte propre au domaine d'application.

Notre but n'est pas de construire directement des dessins d'exploitation mais de mettre à la disposition de l'utilisateur un moyen d'expression à deux (ou trois) dimensions pouvant être dans beaucoup de cas plus utile (car plus parlant) qu'une sortie imprimée. C'est pourquoi nous nous limitons aussi souvent que possible aux schémas. Cela nous permet d'éliminer toute programmation graphique, souvent difficile pour les utilisateurs. Nous nous permettons évidemment la particularisation de ces schémas en utilisant une nomenclature la plus proche possible du domaine d'application concerné (Fig. II-2).

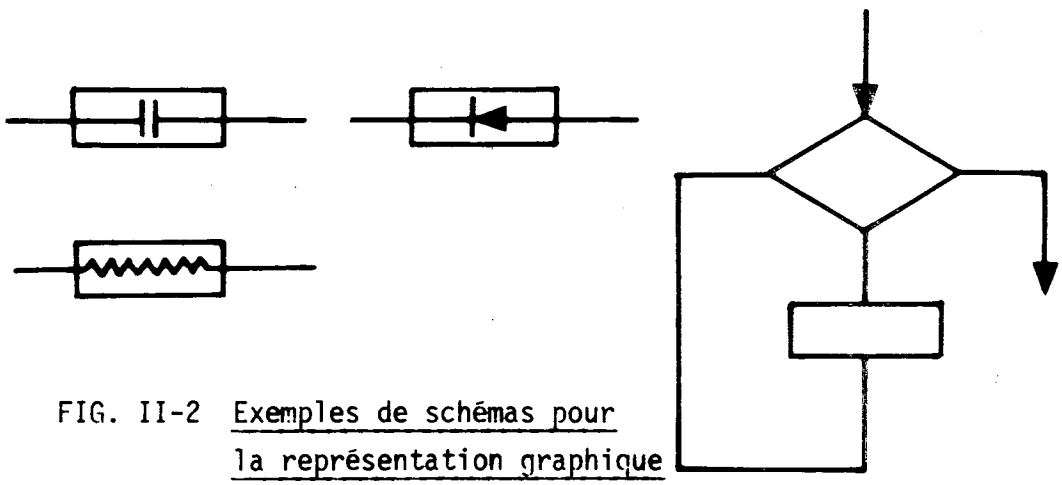


FIG. II-2 Exemples de schémas pour la représentation graphique

Pour des raisons de commodité d'utilisation, il est souhaitable de représenter les données de deux façons d'une part sous forme graphique d'autre part sous forme textuelle (à l'aide d'un langage de description) ces deux formes étant souvent complémentaires. Ceci permet, par exemple, soit d'entrer les données d'un seul coup à l'aide du langage de description, soit de procéder d'une façon

conversationnelle éventuellement graphique à la formulation du problème. Ces deux méthodes doivent, bien entendu, être équivalentes et on peut donc à n'importe quel moment aussi bien sortir le texte du problème sous forme de langage qu'obtenir le schéma graphique correspondant (et éventuellement le modifier). Ceci permet d'assurer dans chacune des phase une DOCUMENTATION AUTOMATIQUE du projet, problème important et pas toujours bien résolu.

Le terminal graphique doit donc permettre l'accès à toutes les données sous une forme la plus appropriée.

-- visualisation du schéma correspondant à un sous-MSR. La visualisation s'effectue en fonction de la phase en cours et selon les exigences de l'utilisateur. C'est-à-dire que par exemple:

  -- lors de l'étape logique on représente les noeuds à l'aide de schémas appropriés et les relations comme des connexions les reliant. On peut évidemment n'afficher que le réseau correspondant à certains types de relation donc une couche du MSR.

  -- lors de l'étape géométrique on peut interpréter les données géométriques en tant que telles: on affiche pour les noeuds les polygones correspondant et on se sert des relations géométriques (jointivité par exemple) pour représenter la répartition décrite.

-- lors de l'étape technique on peut en plus faire apparaître certaines données techniques sur le schéma.

-- visualisation du programme décrivant en langage de description le niveau considéré.

-- visualisation de l'arbre correspondant à la hiérarchisation.

-- visualisation de catalogue choisi.

Pour chaque représentation les commandes autorisées sont celles pouvant être éxécutées avec les données disponibles à cet instant. Il est évident que quelle que soit la représentation choisie une même commande a la même fonction bien que travaillant sur des formats de données différents.

Le module de communication doit assurer les rôles suivants:

-- effectuer l'entrée et les sorties d'informations sous la forme demandée (graphique; textuelle;etc.) et gérer donc le poste de travail.

-- préparer les commandes pour l'unité de traitement; les mettre sous une forme standard indépendante du mode de travail utilisé.

-- effectuer la transformation structure externe =▶ structure interne et remplir donc la banque de données.

-- effectuer la transformation structure interne =▶ structure externe (langage) et vider donc la banque de données.

Le module de communication travaille donc comme un traducteur bidirectionnel rendant d'une part les commandes et les données provenant de l'utilisateur dans une forme standard et d'autre part mettant les données provenant de l'unité de traitement dans une forme souhaitable par l'utilisateur (Fig.II-3).

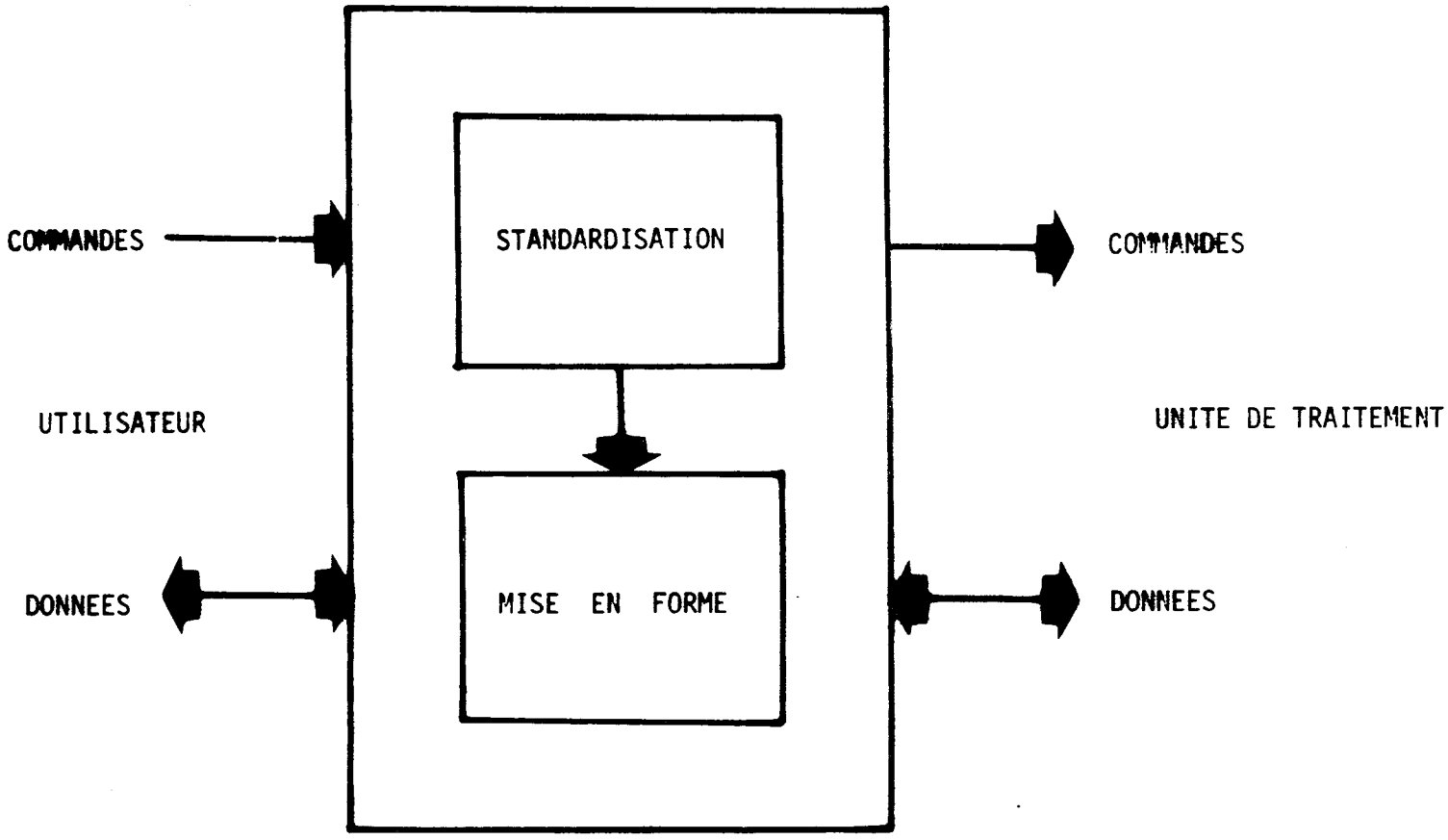


FIG. II-3 Fonctionnement du module de communications

II.1.3 L'unité de traitement

L'unité de traitement exécute les commandes mises sous une forme standard par le module de communication. Ces commandes sont soit les opérations de base, soit les programmes composés à partir des opérations de base, soit les programmes d'application realisés indépendamment du système. (Pour les deux premières il s'agit de l'interpretation, pour la troisième de l'exécution).

Les opérations élémentaires assurent d'une part l'accès (lecture, écriture avec respectivement les resultats et les sources dans la mémoire de travail) à toutes les données disponibles (entités, propriétés, relations) et d'autre part les manipulations des valeurs dans la mémoire de travail.

Nous donnons ici un aperçu des commandes standards du système en respectant la classsification de I.2.4:

-- commandes structurelles:

- -- expansion: deux paramètres: le nom de l'entité et le nombre de niveaux sur lesquels l'opération s'applique: un, plusieurs ou tous.
- -- contraction: le paramètre est la liste d'entités à regrouper.
- -- union, intersection, disjonction: on a deux listes

d'entités comme paramètres.

-- complément: les paramètres sont une entité définissant le sous-MSR et la liste des entités à enlever.

-- commandes d'interrogation:

-- édition d'une entité.

-- édition de la liste des propriétés d'une entité.

-- édition des relations d'une entité.

-- valeur d'une propriété ou d'une relation (contrainte).

-- tri selon un critère (min, max, appartenance, ...) sur une propriété d'une liste d'entités.

-- commandes de modification:

-- annulation ou création d'une entité.

-- annulation ou création de relation ou de propriété.

-- affectation de valeur à un renseignement

-- commandes de contrôle du processus:

-- changement de l'environnement (voir plus loin).

-- changement de phase avec l'activation de toutes les transformations concernées.

-- appel explicite d'une transformation sans changement de phase.

-- commandes de construction d'algorithmes:

-- groupement inconditionnel de commandes (construction de blocs).
-- commande conditionnée par la valeur d'un renseignement ou d'une expression.
-- groupement répétitif de commandes (pour et tant que).
-- groupement de commandes avec le passage de paramètres.

II.1.4 Méthode de travail

Toute conception d'un nouveau système est liée à un problème fondamental: le choix de la "méthode de travail" appropriée. C'est d'autant plus vrai dans le domaine de la C.A.O. que le coût et la puissance de l'outil mis en oeuvre sont importants.

Si l'on envisage une interaction entre l'utilisateur et la machine il faut pouvoir trouver un point d'équilibre où le dialogue peut s'établir, tel qu'il soit profitable à l'un et à l'autre.

Après énumération des différentes possibilités d'utilisation du système, il faut arriver à définir sa méthode de travail.

En effet il nous semble très important et même indispensable de donner des règles strictes quant à l'utilisation de telle ou telle possibilité du système: on minimise ainsi les choix de l'utilisateur

en lui imposant un certain style de travail, ce qui permet au système d'optimiser ses traitements et de ne pas être obligé de se trouver de nouveau dans le même état (standard ou d'équilibre) après chaque commande (ce qui alourdirait considérablement son fonctionnement).

Nous allons décrire les possibilités demandées au système en suivant son fonctionnement.

L'utilisateur formule son problème et pour l'entrer en machine il peut procéder de deux façons: soit à l'aide du langage d'entrée (entrée rapide) soit d'une façon conversationnelle en construisant une description (la plupart du temps graphique).

Pour travailler sur le schéma il doit disposer de quatre types de fonction: orientation, création et manipulation, contrôle, gestion.

-- orientation: fournir les indications sur le schéma (l'emplacement d'un objet demandé, le nom d'un objet désigné, les caractéristiques des objets, etc...)

-- création et manipulation: servir lors de la création et des manipulations des objets selon la phase dans laquelle le système se trouve.

-- contrôle: permettre l'affichage de tel ou tel niveau de hiérarchie, agrandir ou rétrécir une partie, etc...

-- gestion: permettre le changement de format pour les informations: passages langage--▶graphique ou bien graphique--▶langage, sauvegarde temporaire, sortie sur table traçante, etc...

Le nombre de fonctions étant assez important, il faut introduire des groupes d'utilisation: chaque groupe contient des fonctions indispensables à la réalisation d'un certain travail: en imposant ces groupes, ce qui restreint le nombre de fonctions disponibles à un moment donné, nous "guidons" l'utilisateur dans sa démarche de conception. Les groupes correspondent en fait aux fonctions disponibles lorsque le système est dans un état donné.

Nous avons partitionné le système en cinq environnements principaux:

- entrée
- catalogue
- modélisation
- analyse
- sortie

que nous allons passer en revue.

Environnement d'entrée

Initialisation du système et du terminal graphique et prise en compte des données de l'utilisateur. Ces données peuvent être de trois sortes:

-- description de la configuration du matériel sur lequel on va travailler (terminal, terminal graphique, table traçante, etc...)
-- spécification des catalogues utilisés pendant les traitements.
-- description du problème.

Pour cette dernière trois cas se présentent:

-- l'utilisateur désire un travail uniquement conversationnel: il n'y a pas de description initiale.
-- l'utilisateur veut reprendre une description précédemment créée et sauvegardée sous forme interne; on devra la charger.
-- l'utilisateur a préparé, à l'aide du langage de description, une nouvelle description; il faut la traduire en forme interne.

Remarque: dans le troisième cas, pour constituer une représentation utilisable pour une sortie graphique il faut, si la description ne les contient pas (description logique), induire des données géométriques. Dans le chapître III nous décrirons cette démarche dans un contexte particulier.

Environnement de catalogue

Il a été introduit pour distinger les objet-valeurs utilisés comme valeurs des renseignements; les raisons en sont:

-- obliger l'utilisateur à prendre conscience de la définition d'un nouvel objet ce qui l'incite à ne pas les multiplier.
-- contrôler les manipulations des objets partagés.
-- disposer à tout instant de la liste de tous les objets utilisés, avec leurs d'occurences.

Environnement de modélisation

C'est dans cet environnement que se déroule le processus de conception. Pour cela on procède à la construction et à la modification du MSR correspondant à l'objet.

Toutes les données sont disponibles mais les modifications des données partagées se limitent aux spécifications quantitatives (actualisation).

Remarque: lors du travail sur une entité partagée il faut indiquer la portée des opérations: selon qu'elles s'appliquent à une ou à toutes les occurences nous parlerons d'opérations locales ou globales.

Environnement d'analyse

Il sert à porter des jugements (selon divers critères) sur le prototype.

On distingue jugements statiques (sur la structure) et dynamiques (sur le fonctionnement). Ces traitements sont essentiellement spécifiques à chaque application et souvent ils nécessitent, pour des raisons d'efficacité, les structurations particulières effectuées sous forme de transformations.

Environnement de sortie

Il regroupe tous les traitements effectuant les changements de représentation et de support (en sortie): sortie sur table traçante, sauvegarde d'un prototype, génération d'une description en langage ou tout simplement sortie du système.

Les deux schémas qui suivent permettront de mieux saisir le fonctionnement du système.

L'enchaînement des différents environnements peut être schématisé comme suit (Fig. II-4):

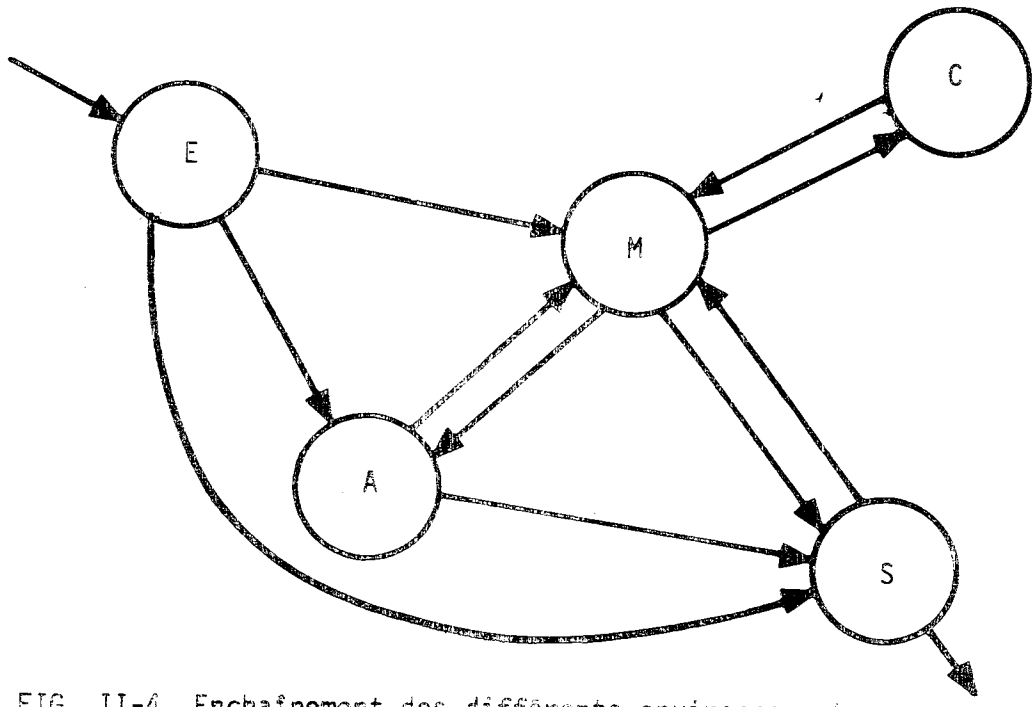


FIG. II-4 Enchaînement des différents environnements

Commentaires:

-- le passage E⟶S représenterait ce que l'on appelle la documentation automatique. Dans ce cas l'utilisateur n'a besoin ni du terminal graphique ni des possibilités conversationnelles du système. Il paraît intéressant de construire un sous-système n'ayant que ces possibilités, qui ne demande pas de ressources superflues.

-- le passage E⟶A⟶S représente l'analyse et la documentation de l'objet pouvant aussi être effectués par lot.

-- le passage par M représente un travail de conception essentiellement conversationnel.

Dans chaque environnement les commandes ne peuvent travailler que d'une certaine manière sur les données disponibles (données du problème et les catalogues). Les actions autorisées dans chaque environnement sont les suivantes (Fig.II-5):

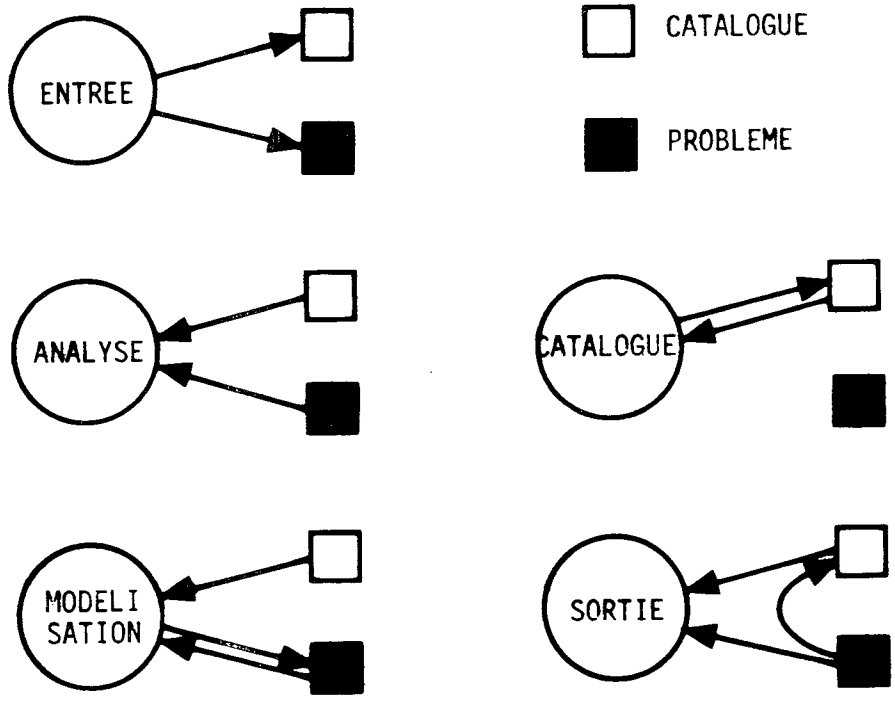


FIG. II-5 Les actions sur les données dans chaque environnement

Commentaires:

-- entrée: remplissage seulement

-- analyse: utilisation seulement

-- cataloque: modifications des catalogues

-- modélisation: utilisation des catalogues et modification du problème

-- sortie: utilisation des catalogues et du problème et eventuellement cataloquage du problème

II.2 Utilisations du système

II.2.1 Différents types d'utilisation

Même si le formalisme MSR est très simple du point de vue de sa structure, il peut paraître trop abstrait, trop puissant et trop lourd pour pouvoir être utilisé directement par les utilisateurs. C'est pour rendre le travail de chacun aussi efficace et agréable que possible que nous considérons que le système SIGMA-C.A.O. s'adresse aux trois types d'utilisateurs:

-- les concepteurs de systèmes spécialisés qui construisent une machine particulière pour une application donnée.

-- les utilisateurs "ordinaires", qui se servent du système pour résoudre leurs problèmes spécifiques en utilisant les outils offerts par le système.

-- les utilisateurs "évolués", qui ne se contentent pas des outils disponibles et veulent faire évoluer les possibilités du système selon le schéma d'Alexander(1) Fig.II-6.

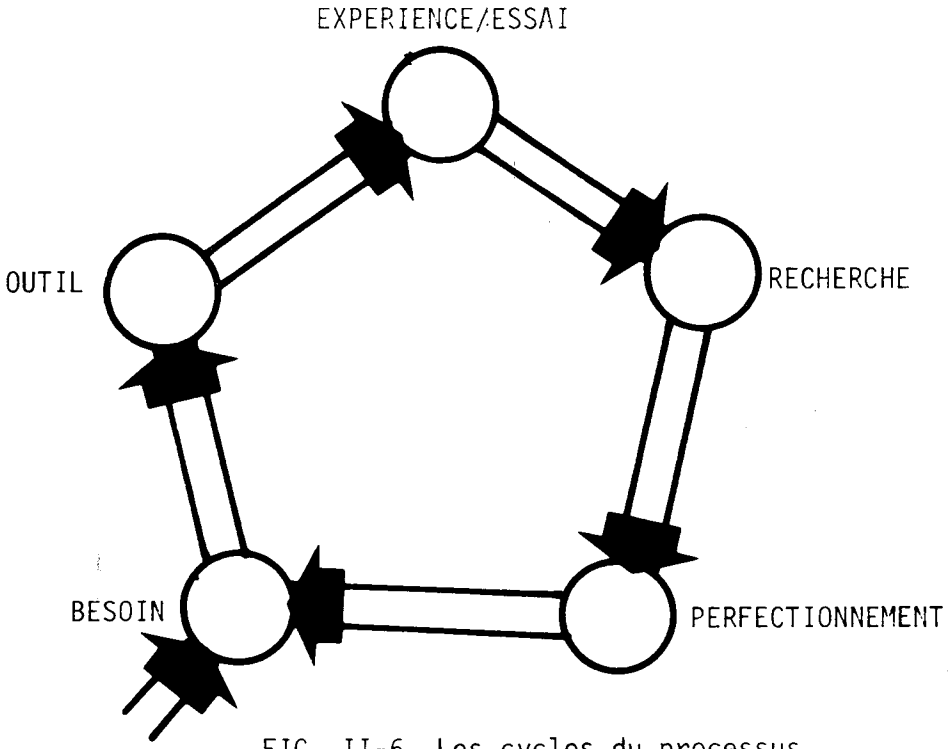


FIG. II-6 Les cycles du processus de conception

La particularisation de la machine dépend beaucoup du projet que l'on se propose d'étudier. Selon que le projet est ouvert ou fermé on peut plus ou moins bien prévoir les exigences que l'on aura sur le système.

Les MSR peuvent être particularisés d'une façon plus ou moins sévere. Pour les projets fermés on peut prédéfinir complétement toute la structure des données et les traitements. On peut donc acceder aux données en évoquant les caractéristiques dont on peut fournir ou obtenir les valeurs. Dans ce cas l'utilisateur n'a connaissance que de son problème. Pour le résoudre il n'a à aucun moment à sortir de son domaine et n'est pas obligé de tenir compte de la représentation

informatique comme par exemple la spécification informatique des objets (entier, réel, structure, tableau,...etc..) qui lui reste complétement transparente.

Si l'utilisateur veut ou doit, notamment pour les projets ouverts, faire évoluer la structure et le fonctionnement du système, il doit évidement connaître au moins en partie l'organisation informatique de son projet.

II.2.2 Construction d'un système particulier

La conception d'un nouveau système s'effectue en plusieurs étapes. Tout d'abord il faut formuler précisement le but du système, puis analyser de quelle façon on va l'atteindre. C'est-à-dire déterminer les données du problème, leurs liens et dépendances pour déterminer les données indispensables devant être fournies par l'utilisateur et les différents traitements et leur enchaînement pour obtenir les résultats demandés.

Une fois l'aspect fonctionnel défini, on peut etudier l'aspect organique. Pour cela on procéde à la structuration effective des données et on détermine les traitements sur ces données. C'est là que l'on doit tenir compte des facteurs humains et economiques: choix de langage d'intervention de l'utilisateur (langage de description et de commande), coût en temps de calcul ou en place mémoire.

La particularisation du système SIGMA-C.A.O. pour un probleme s'effectue de la façon suivante:

On spécialise le MSR pour ce problème en définissant les structures que l'on veut prendre en compte et on spécifie la sémantique liée à ce support structurel en déclarant les renseignements; éventuellement on décrit les entités particulières servant de base pour le problème et on définit les phases de résolution et le graphe d'enchaînement.

Déclarer un renseignement veut dire préparer un squelette portant comme nom le type de renseignement (nom générique), dans lequel on définit toutes les caractéristiques (type, information, classe, transformation, attribut, orientation).

Ayant la description de tous les renseignements on peut générer les fonctions d'acces particularisées pour le probleme et éventuellement construire les traitements plus complexes et significatifs pour l'utilisateur (fonctions de base).

Ces informations sont stockées dans un cataloque et fournissent donc la spécialisation du système pour un problème particulier.

Pendant la résolution d'un problème de la famille pour laquelle le MSR a été particularisé on procéde au remplissage de renseignements predeclarés:

De cette façon on construit le MSR particulier, puis on applique les traitements pour faire avancer la résolution soit en faisant intervenir un nouveau renseignement dans le MSR soit en modifiant les informations d'un renseignement déjà présent (un simple changement de valeur).

Pour faire intervenir un nouveau renseignement on prend une copie du squelette catalogué du renseignement (désigné par le nom générique), on lui associe les informations voulues, éventuellement on lui donne un nom propre et s'il s'agit d'une propriété on la place dans une entité ou s'il s'agit d'une relation on établit les connexions la liant aux entités.

II.2.3 Exemple: circuits électroniques

Essayons de montrer comment on pourrait particulariser le système SIGMA-C.A.O. pour qu'il permette la manipulation et la simulation des circuits électroniques. Il s'agit d'un exemple volontairement simplifié et nous nous inspirons d'IMAG 3(21) pour ce qui concerne l'analyse du problème et sa mise en oeuvre informatique.

La simulation de circuits élécroniques est un problème fermé, on peut donc déterminer à l'avance tout le comportement du système.

Prenons un exemple (Fig. II-7):

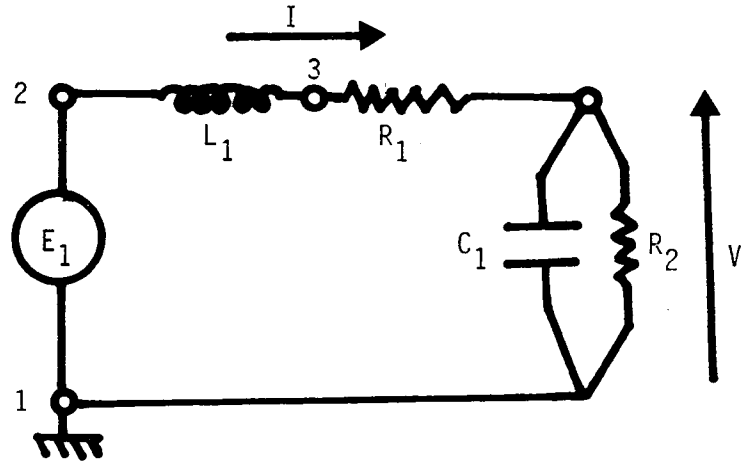


FIG. II-7 Exemple de circuit électrique

On schématise ce réseau par un graphe avec deux types de sommets: des sommets composant et des sommets noeud. Le problème est de déterminer les tensions entre les noeuds et les intensités dans les composants.

Le comportement des composants est décrit par les équations caractéristiques indiquant leur influence sur les intensités et les tensions.

Souvent pour des raisons de commodité d'emploi on groupe les composants élémentaires en macrocomposants (exemple C1et R2 Fig. II-7). Les macrocomposants peuvent être soit décrits par la composition de composants élémentaires soit directement en fournissant les équations les caractérisant.

Lors de la description d'un réseau on place les composants en indiquant les connexions avec les nœuds et on fixe les valeurs des paramètres intervenant dans les équations caractéristiques.

Le système d'équations correspondant au réseau étudié est constitué d'une part par les équations caractéristiques des composants et d'autre part par les équations représentant la topologie du réseau (équations de Kirchhoff). Ces dernières peuvent être déduites du réseau.

Les objets que le système doit traiter sont d'une part les composants élémentaires pour lesquels on a la structure unique suivante:

- nom (4 caractères)
- nature (1 car. R,L,...)
- liste des noeuds (2 nœuds)
- valeur (éventuellement expression)

et d'autre part les composants complexes pour lesquels la structure n'est pas complètement figée:

- nom (4 car.)
- nature (plusieurs caractères)
- liste de noeuds (plusieurs nœuds)
- valeurs (plusieurs valeurs)
- équation caractéristique (optionnelle)

L'utilisateur doit déclarer les composants complexes (macrocomposants) avant de les utiliser dans le réseau. En fait il se construit un cataloque de composants contenant les composants élémentaires et les composants complexes effectivement déclarés. A ce catalogue il peut adjoindre les bibliothèques de composants précédemment déclarés et concervés pour leur intérêt certain.

Lors de la construction il choisit un composant dans le catalogue ou dans une bibliothèque et il l'implante en le liant aux noeuds et en spécifiant les valeurs des paramètres.

Sur le réseau construit il peut effectuer les différents calculs correspondant à la simulation du comportement du circuit.

Si nous regardons plus en détail le processus de résolution nous distinguons quatre phases:

-- définition d'un macrocomposant (DM)

-- construction d'un réseau (CR)

-- choix du mode de simulation (CS)

-- sorties de résultats (SR)

Leur enchaînement est défini comme suit (Fig.II-8):

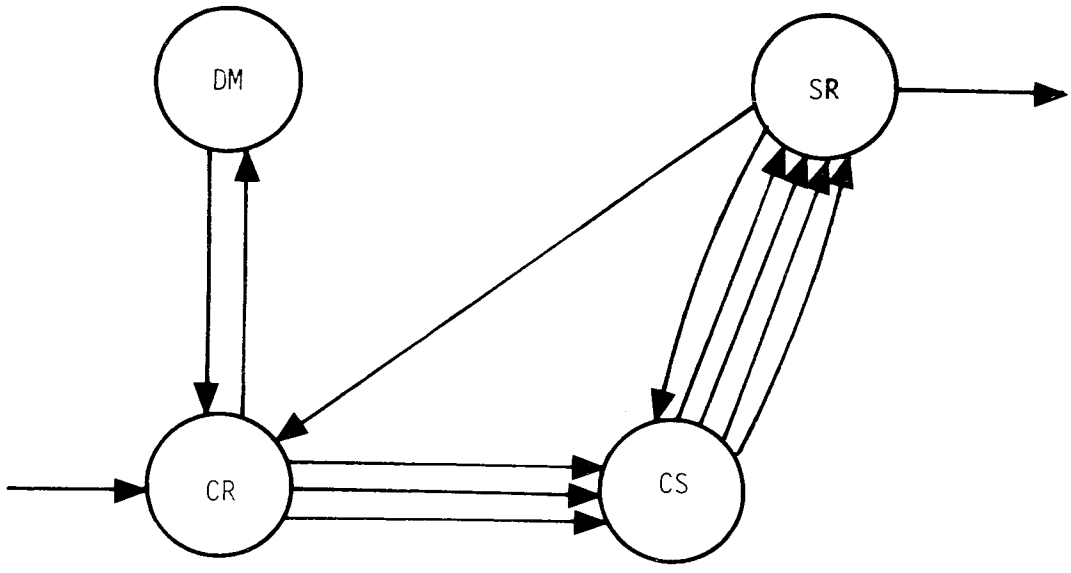


FIG. II-8 L'enchaînement de phases dans l'IMAG 3.

Commentaires:

-- les passages de CR à CS représentent la génération d'équations et leur mise en ordre et les outils d'exploitation correspondant au mode d'utilisation choisi (continu, alternatif, transitoire).

--les passages de CS à SR représentent les calculs proprement dit.

Pour réaliser avec SIGMA-C.A.O. cette application on définit les phases et leur enchaînement selon la Fig.II-8 et on prédéclare les squelettes des renseignements indispensables pour chaque phase:

Dans les phases DM et CR on travaille sur les composants élémentaires ou complexes, les noeuds et les relations nécessaires pour la construction du réseau.

Le passage de CR à CS fait appel à une transformation qui a pour but de générer la nouvelle représentation en construisant les équations caractéristiques, les équations de Kirchhoff et en procédant à leur mise en ordre.

Dans les phases CS et SR on travaille uniquement sur ces équations.

L'utilisateur définite les actions qu'il souhait pouvoir entreprendre dans chacune des phases:

-- DM: définition d'un macro composant
-- CR: construction du réseau, choix du mode d'exploitation
-- CS: définition de points de mesure, paramétrage du calcul, choix du calcul
-- SR: exploitation des sorties, choix du prochain travail

Ces actions sont implémentées comme commandes du système spécialisé et devienent automatiquement disponibles quand on se trouve à la phase concernée.

II.2.4 Méthodes et outils de construction

Le travail de l'implémentation d'un système particulier peut être effectué de différentes façons dont deux extrêmes sont les suivantes:

-- la méthode directe: après une analyse détaillée du problème on définit le système particulier et on procéde directement à son implémentation.

-- La méthode par prototype: après une analyse du problème on implémente le prototype fonctionnellement équivalent au futur système mais réalisé avec les outils facilitant la mise en oeuvre. Cette méthode permet une vérification rapide du comportement fonctionnel et son raffinement ainsi que la définition organisationnelle définitive. Cela n'est qu'après avoir obtenu un comportement satisfaisant du prototype que l'on procéde à une implémentation définitive.

Il va de soi que cette deuxième approche nous paraît plus intéressante car elle permet l'utilisation de techniques C.A.O. à un autre niveau.

Lors de la définition d'un système particulier il s'agit de spécifier les traitements que le système doit effectuer, les données sur lesquelles les traitements vont travailler et le dialoque que l'utilisateur pourra avoir avec le système.

Les techniques C.A.O. peuvent apporter une aide considérable dans la réalisation de certaines parties d'un tel système et notamment:

La technique de prototype introduite précédemment et permettant une définition progressive et itérative de la maquette du système (elle sert surtout à obtenir une bonne structure de données et les traitements efficaces) peut être couplée à la génération d'un système performant correspondant à la maquette satisfaisante.

Les dialogues avec l'utilisateur se font très souvent à l'aide d'un langage de commande qui lui doit être proche. Le générateur de langages de commande peut faciliter cette tâche et permet de fournir un langage de commande souhaité.

Les problèmes à résoudre peuvent être présentés et rendus sous une forme claire, parlante et explicite grâce au langage de description. La technique de génération de langages de description peut fournir un langage de description approprié et son traducteur-vérificateur.

Certaines parties du système peuvent être prédéfinies sous forme de modèle et leur utilisation dans un système particulier se fait par actualisation. Le modèle pour les manipulations de schémas en est un exemple.

II.2.5 Maquette de l'application

La conception par maquette d'un système particulier permet d'aboutir itérativement à un système fonctionnellement satisfaisant, puis d'en dégager la structure organique.

Un système est caractérisé par:

-- les phases de conception et leur enchaînement

-- les traitements intervenant dans et entre les phases

-- les structures de données pour chacune des phases

Pour réaliser un système particulier il faut d'une part définir ces trois constituants, d'autre part les lier. La maquette peut simplifier ce travail en fournissant les moyens facilitant ces deux tâches:

-- en enregistrant toutes les définitions contenues dans les trois constituants on matérialise la structure du système.

-- en interprétant les liens entre les différents composants on permet le changement de toute partie sans nécessité de modification des autres.

Cela peut être réalisé car on n'utilise pas de fonction d'accès particulières et directes, mais des fonctions générales et indirectes qui, avant d'accéder à l'objet demandé, se renseignent (en interprétant la déclaration enregistrée le concernant) sur ses caractéristiques (nature, emplacement, etc...).

La réalisation en est la suivante (Fig II-9):

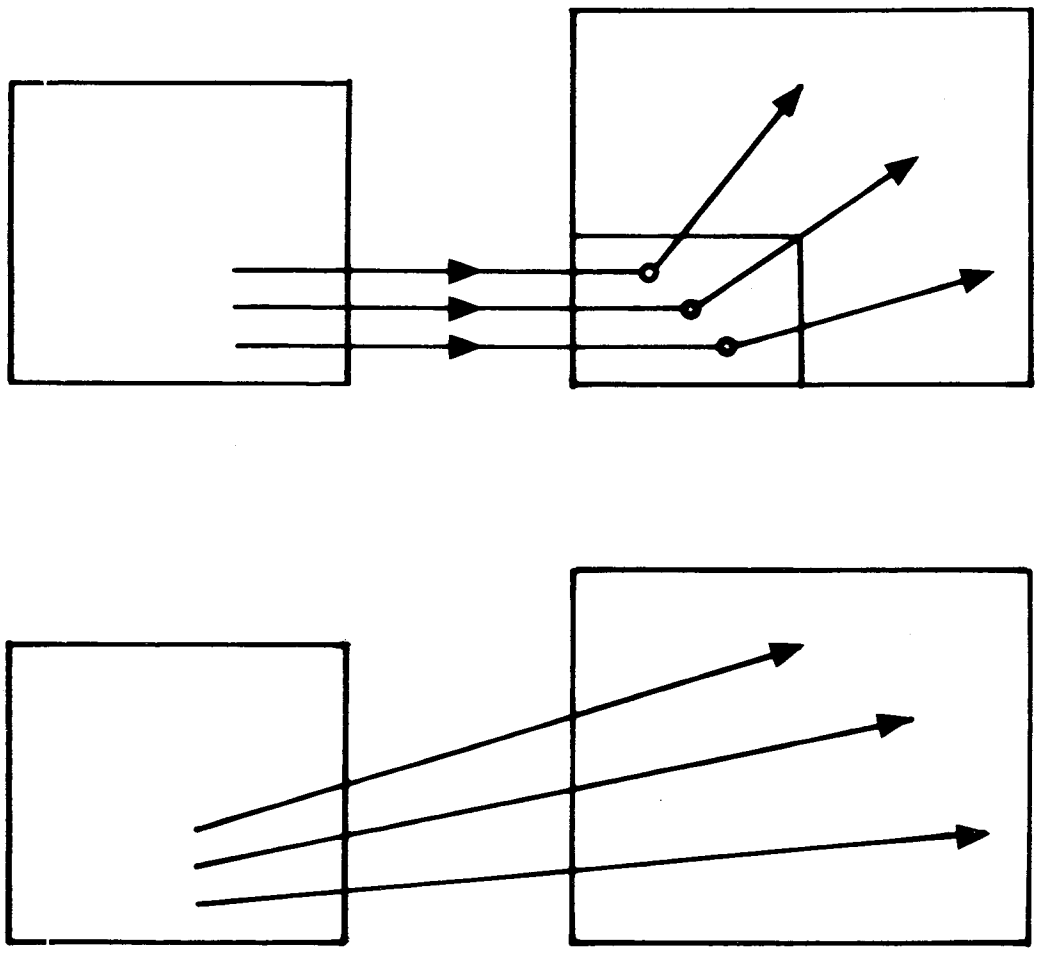


FIG. II-9 Fonctionnement par maquette et le fonctionnement réel

une partie du système contient la description de sa structure. Cela permet des accès aux données même si on ne connait pas leur implémentation exacte.

Le MSR va donc contenir, une fois pour toute, les objets particuliers ayant une structure figée, qui ont pour but de recevoir toutes les définitions des constituants; dans une partie du système les fonctions d'accès pourront donc se renseigner sur la structure de l'autre partie, concernée par le problème particulier.

Pour le fonctionnement du système la méthode du prototype est un frein ; c'est pourquoi dès que son aspect fonctionnel est satisfaisant il faut produire une version rapide, en remplaçant les fonctions d'accès général par les fonctions particulières et l'interprétation par l'éxécution.

On peut ainsi, en abandonnant toutes les parties non directement utiles, fournir un système très particulier, figé mais efficace et peu volumineux, pouvant loger sur une petite machine. Schématiquement il s'agit d'aplatir la structure présentée sur la Fig I-3 b et de n'en garder que les modules utiles. Ce travail peut aussi, d'après nous, être, au moins en partie, fait automatiquement.

II.2.6 Langages de commande

Un système interactif doit fournir à l'utilisateur des moyens de dialogue; on les appelle en général les langages de commande.

Dans un langage de commande il est souhaitable d'utiliser aussi souvent que possible les moyens apportés par les terminaux graphiques, surtout le photostyle et les touches de fonctions. Cela permet d'avoir un dialogue relativement clair (utile pour les non spécialistes) car l'affichage d'un texte, même s'il est long, est instantané; ce dernier point est un avantage sur les terminaux alphanumériques classiques pour lesquels il faut avoir deux versions de dialogue: l'une explicite et relativement longue, utile à l'initiation, l'autre avec un minimum d'indications, rapide, mais peu explicite.

L'utilisation des touches de fonctions et du photostyle amène souvent à concevoir un langage constitué de commandes plus ou moins simples et travaillant d'une façon autonome: à chaque touche est attachée une certaine fonction; le déclenchement de la touche provoque l'affichage dans la zone système du format exact de la fonction, avec indication des différents choix possibles. On a un dialogue convenablement guidé, qui permet de détecter facilement des commandes erronées que l'on peut interdire.

On peut constater que le problème de la construction d'un langage de commande pour un système particulier a été beaucoup étudié et que les techniques de génération de langage de commande (5), (17) permettent une construction rapide de langages spécialisés.

En partant d'une approche purement conversationnelle, on se rend compte de la nécessité d'introduire, des possibilités de groupement de commandes, pour construire des fonctions plus puissantes qui rendent le travail moins pénible.

C'est pourquoi il faut permettre la constitution de macro-commandes avec des boucles, des expressions conditionnelles, éventuellement des séquences algorithmiques gérant des résultats partiels et d'autres offrant la possibilité de changement programmé d'environnement.

Ceci ne diminue en rien la valeur de l'approche car l'utilisateur obtient ainsi une possibilité de contrôle plus ou moins fin.

II.2.7 Langages de description

Pour certains problèmes de conception il faut entrer et sortir des informations variées mais pouvant être bien structurées. Un langage de description permet d'avoir ces informations sous une forme claire et significative. Pour qu'il soit facilement interprétable par un non informaticien, il doit être simple et relativement proche du domaine d'application.

Le problème de la conception d'un langage de description particulier est donc posé. Si nous regardons par exemple les langages de description de CASSANDRE (7) (uniquement la partie structurelle), d'IMAG (21) ou celui développé par RECHENMANN (11) pour la modélisation socio-économique, on peut constater certaines similitudes.

Il nous paraît intéressant d'étudier la possibilité de concevoir un générateur de traducteur-vérificateurs de langages de description.

Le problème nous paraît abordable pour les raisons suivantes:

-- il s'agit d'une description statique (voir la remarque dans I.3.1)

-- les problèmes traités sont relativement semblables

-- la formulation s'inspire du formalisme MSR mais n'en a pas sa rigueur, car ce qui est indispensable pour une description avant tout orientée vers l'utilisateur, c'est surtout la commidité d'expressicn

-- on fait la traduction dans une structure particulière issue du formalisme MSR

On pourrait donc définir une structure unique, c'est-à-dire une syntaxe qlobale pour toute la famille de langages. On traitera les particularités, pour partie dues à la commodité d'expression, pour partie dues au remplissage de la structure, sous forme de paramètres de génération.

Le fonctionnement des générateurs de traducteur-vérificateur de langage de description peut être schématisé comme suit:

pour prendre en compte un langage spécialisé il faut indiquer les particularités qui lui sont propres. Par exemple option par défaut pour les déclarations, syntaxe exacte des instructions etc... Ceci s'effectue par un langage de particularités des langages de description (LPLD).

La structure qlobale est décrite une fois pour toute par le langage de structuration générale (LSG) et elle correspond au MSR général.

Le couple LPLD + LSG assure la traduction du langage particulier de description (LDi) dans une structure générale.

Pour stocker les informations dans le MSR particulier (MSR i) il faut aussi décrire ces particularités: c'est le langage de particularités de la structure d'accueil (LPSA).

Le couple LSG + LPSA assure la traduction dans la structure particulière MSR i.

Le fonctionnement général pour lequel il serait souhaitable qu'il permette les traductions dans les deux sens peut êtreschématisé de la façon suivante:

LDi ⟷ LPLD + LSG + LPSA ⟷ MSRi

La structure d'un traducteur-vérificateur particulier est donc liée à un LDi et à un MSRi; on peut la schématiser sur la figure II-10.

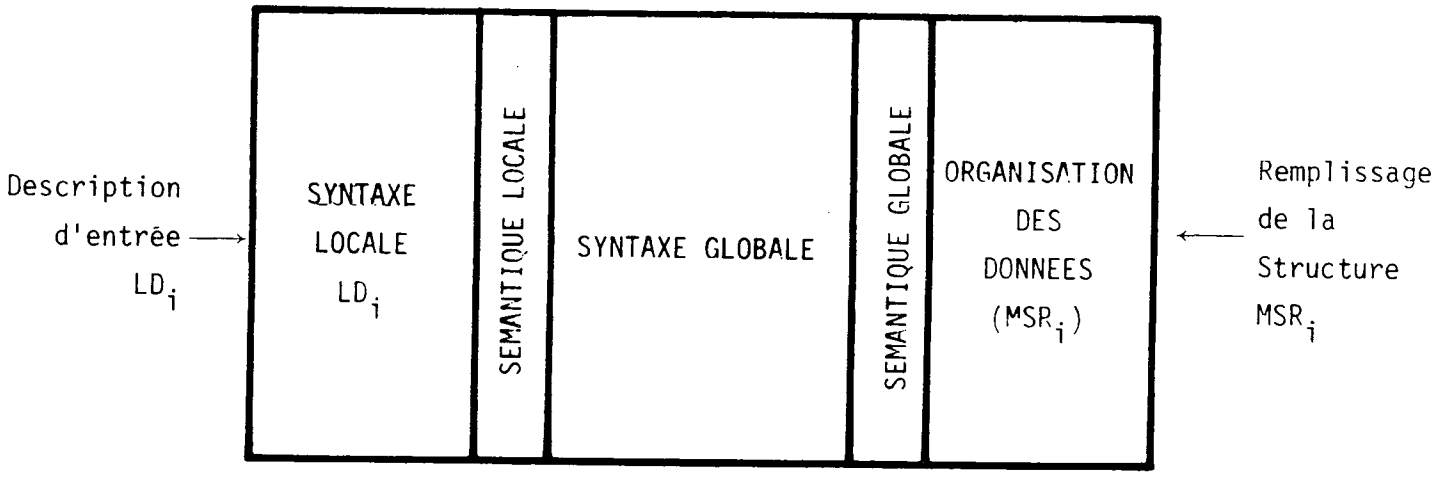


FIG. II-10 Structure générale du traducteur vérificateur pour les langages de descriptio

II.2.8 Modèle pour les manipulations de schémas

La représentation graphique sous forme symbolique est très courante dans le processus de conception (circuits logiques; circuits électroniques; molécules chimiques, etc...). Il nous a paru intéressant d'étudier le modèle pour les manipulations graphiques d'une telle représentation.

Essayons d'abord de voir en quoi consiste le problème.

Par exemple: quand un électronicien travaille graphiquement, il lui est naturel de le faire sur les composants (transistors, diodes, capacités, ...), on doit remarquer que le schéma seul n'est pas suffisant: il faut évidemment en saisir son aspect fonctionnel pour pouvoir effectuer les analyses demandées.

Dans le schéma il faut donc faire la différence entre:

-- le dessin du schéma

-- les paramètres associés faisant le lien entre l'aspect schématique et l'aspect fonctionnel

-- les points d'attache permettant de lier différents schémas

Le formalisme MSR permet grâce à la notion de couche de faire cohabiter différentes descriptions d'un problème; par exemple: description graphique et description fonctionnelle.

On peut donc étudier indépendamment les deux parties, ce qui nous permet de considérer le MMS comme un MSR particulier (avec sa structure et ses traitements propres).

Nous allons distinguer trois aspects fondamentaux du modèle:

-- la structure logique des données sur laquelle le modèle est basé

-- la structure de fonctionnement (comportement dans chaque environnement)

-- les fonctions disponibles dans chaque environnement

Structure logique de données

Dans la structure il faut d'une part assurer l'obtention du graphique (ce qui est effectué par le générateur de schémas, qui est catalogué et propre à chaque application) et d'autre part la structure effective des données qui doit contenir, pour un schéma, son type, les paramètres intervenant dans la génération de schéma (dimension, nombre d'entrées sorties, etc...), les paramètres fonctionnels associés au

schéma et pouvant être manipulés graphiquement, les points d'attache servant à lier le schéma aux autres schémas (donc réalisant les connexions entre schémas).

Le fonctionnement étant défini dans II.1.4, précisons le contenu de chaque environnement:

-- ENTREE: spécification des catalogues: générateur de schéma, bibliothèque de schémas de base et de schémas complexes

-- MODELISATION: travail de manipulation (voir les fonctions plus loin)

-- CATALOGUE: mise à jour des catalogues de schémas, c'est-à-dire travail de manipulation sur les schémas catalogués

-- ANALYSE: lecture et modification de valeurs de paramètres fonctionnels

-- SORTIES: différents types de sorties de schémas

Fonctions de manipulation disponibles dans l'environnement de modélisation: nous pouvons y distinguer quatre classes:

-- orientation sur le schéma
-- orientation dans le catalogue
-- construction d'un schéma
-- structuration

que nous allons passer en revue.

ORIENTATION SUR LE SCHEMA

Nous distinguons trois types de fonction d'orientation:

-- les fonctions d'affichage et de changement d'affichage permettant d'obtenir: le schéma correspondant au niveau choisi dans la description hiérarchique de schémas, l'arbre d'emboîtement de ces niveaux et le changement du niveau visualisé

-- les fonctions d'orientation graphique permettant de trouver l'emplacement de toutes les entités et de leurs relations

-- les fonctions d'orientation sur les caractéristiques fonctionnelles des entités donc sur les propriétés et sur les connexions

ORIENTATION DANS LE CATALOGUE

Permet de visualiser, page par page, les objets du catalogue choisi pour permettre le choix de l'objet.

CONSTRUCTION D'UN SCHEMA

Il s'agit de construire le schéma avec les objets choisis dans les cataloques donc placer, déplacer, enlever les objets et les lier avec les autres et fournir les caractéristiques fonctionnelles.

STRUCTURATION

Nous distinquons les structurations verticales et horizontales. Pour les structurations verticales il s'agit de constituer un schéma complexe à partir d'un schéma en cours de construction, ou, au contraire, permettre la décomposition d'un schéma complexe. Les structurations verticales (union, intersection negation) interviennent lors de la composition horizontale.

Il est intéressant de voir comment on peut lier un langage de description avec le modèle de manipulations symboliques.

Nous ferons la distinction entre deux cas:

-- l'aspect topologique est contenu dans le langage de description; le traducteur vérificateur, en fournissant la structure interne, produit toutes les informations nécessaires, et pour une utilisation graphique il suffit de disposer d'un cataloque de schémas correspondant au problème étudié.

-- l'aspect topologique n'est pas contenu dans le langage de description (circuits logiques par exemple); il faut donc déduire de la description les données topologiques et effectuer une répartition spatiale pouvant être exploitée sous forme graphique. Nous verrons dans le chapitre III la mise en oeuvre d'une telle démarche.

II.3 Implémentation de la machine abstraite

Dans ce paragraphe nous discuterons des differentes possibilités d'implémentation de la machine abstraite. Les deux aspects fondamentaux de ce problème sont l'organisation des traitements et le choix de la gestion des données.

Nous pensons que l'aspect gestion des données offre beaucoup plus de variétés que l'aspect traitement; c'est pourquoi nous présenterons les trois méthodes types, en discutant de leurs avantages respectifs.

II.3.1 Par fichiers

La gestion des données par fichiers n'est pas considérée comme élégante; mais cela ne doit pas vouloir dire que c'est, à priori, forcément une mauvaise solution. Il est vrai qu'elle oblige le système à s'occuper dans la partie traitement de la gestion des fichiers; ce qui peut être gênant. Mais il faut analyser en détail les types de traitements (les données en entrée et en sortie) et les types de données que le système gère pour pouvoir porter un jugement valable.

Si les traitements sont cohérents et permettent de travailler sur un nombre restreint d'ensemble de données toujours identiquement structurées, une telle solution est excellente (Fig II-11a). Nous pouvons donner comme exemple l'organisation des fichiers dans le système CASSANDRE (voir chapître III).

A partir du moment où les traitements nécessitent ou produisent des ensembles de données variés, on doit inclure dans les traitements des préparateurs ayant pour rôle de mettre en forme, ou produire un fichier correspondant au traitement souhaité (Fig II-11b).

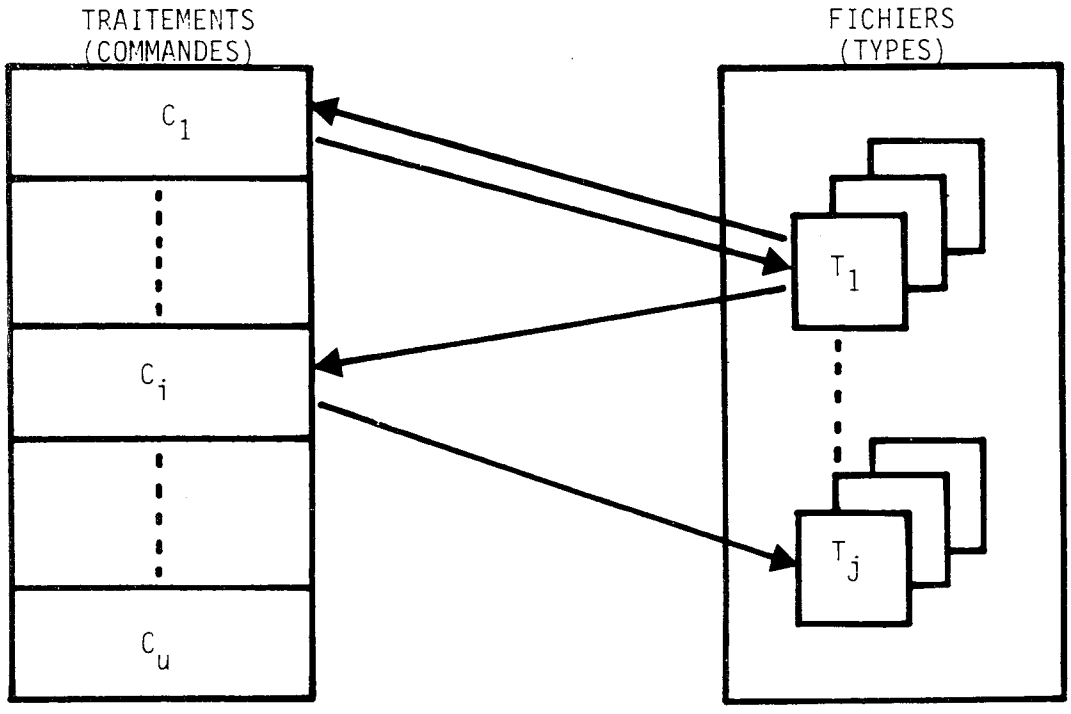


FIG. II-11-a Organisation par fichiers

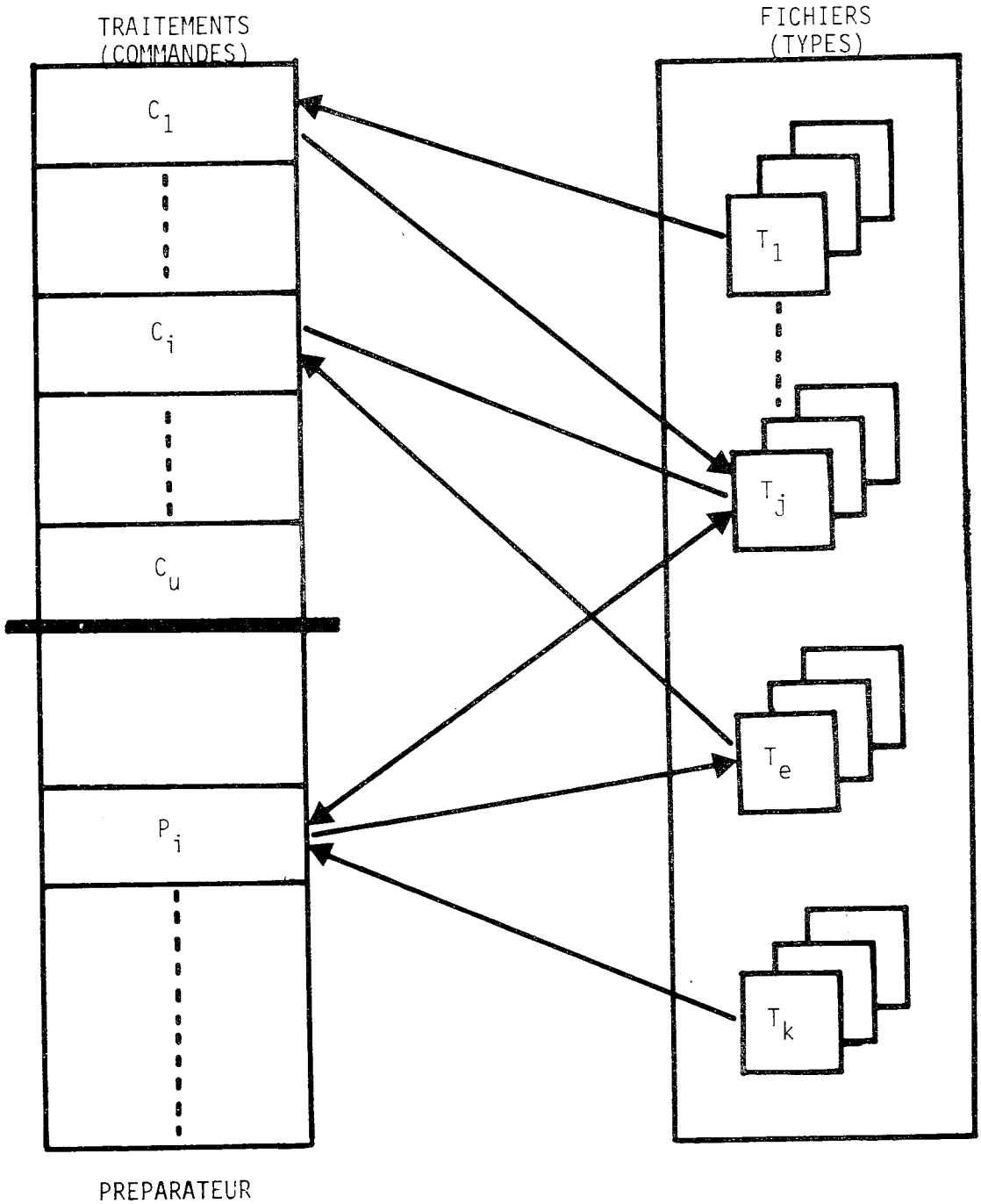


FIG. II-11-b Organisation par fichiers

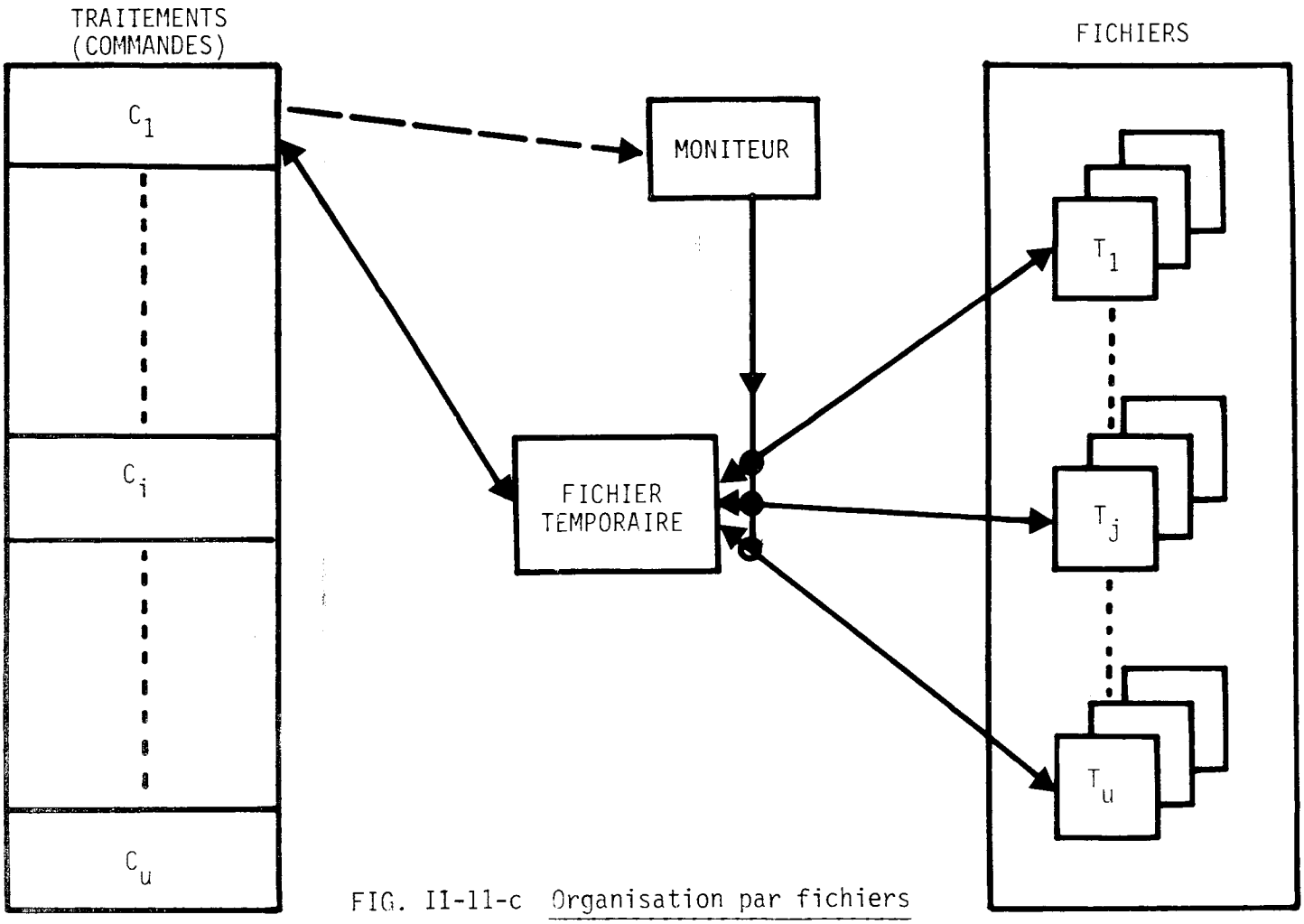


FIG. II-11-c Organisation par fichiers

Si la variété des demandes est très grande, on peut envisager de garder les informations sous une forme standard et remplacer les préparateurs par un moniteur standard. Son rôle est de fournir à chaque traitement le fichier correspondant et remettre les resultats produits sous la forme standard (Fig.II-11c).Il est evident que l'on peut considérer une telle solution comme simulation rudimentaire d'une base de données.

II.3.2 Par base de données

La gestion de données par une base de données est souvent considérée comme une solution idéale. Evidemment la gestion est automatique et les données sont disponibles directement (Fig II-12); mais le temps d'accès peut être considérable, il peut donc y avoir gêne pour des traitements complexes et répétitifs.

Néanmoins cette solution nous paraît la seule valable pour le fonctionnement du système SIGMA-C.A.O. comme maquette d'une application. Mais il est plus que souhaitable d'avoir eu moins une amélioration du temps d'accès pour des accès répétitifs à une même donnée.

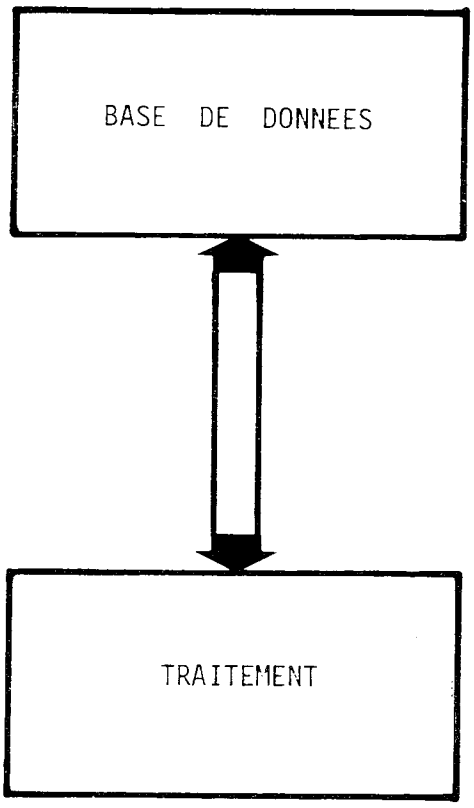


FIG. II-12 Organisation par base de données

II.3.3 Par base de données et fichiers

Cette méthode peut réunir les avantages des deux méthodes précédentes et, en plus, elle s'avère très utile: en effet, non seulement elle rend plus efficaces les traitements répétitifs travaillant sur un ensemble restreint de données, mais surtout elle peut assurer la communication entre les environnements de traitement d'informations et de stockage d'informations. Une telle séparation est courante dans les systèmes de bases de données et nous l'avons rencontrée quand nous avons voulu implémenter le système SIGMA-Archi (chapitre IV) à l'aide du système SOCRATE. Nos traitements étant très variés et écrits dans différents langages de programmation (FORTRAN, ALGOL W, PL/I, assembleur, LISP) nous avons été amenés à travailler en dehors du système SOCRATE. Le passage de données se fait justement par fichier. Dans SOCRATE nous avons un moniteur qui selon la demande crée le fichier et rentre dans la base les informations produites par le traitement (Fig II-13).

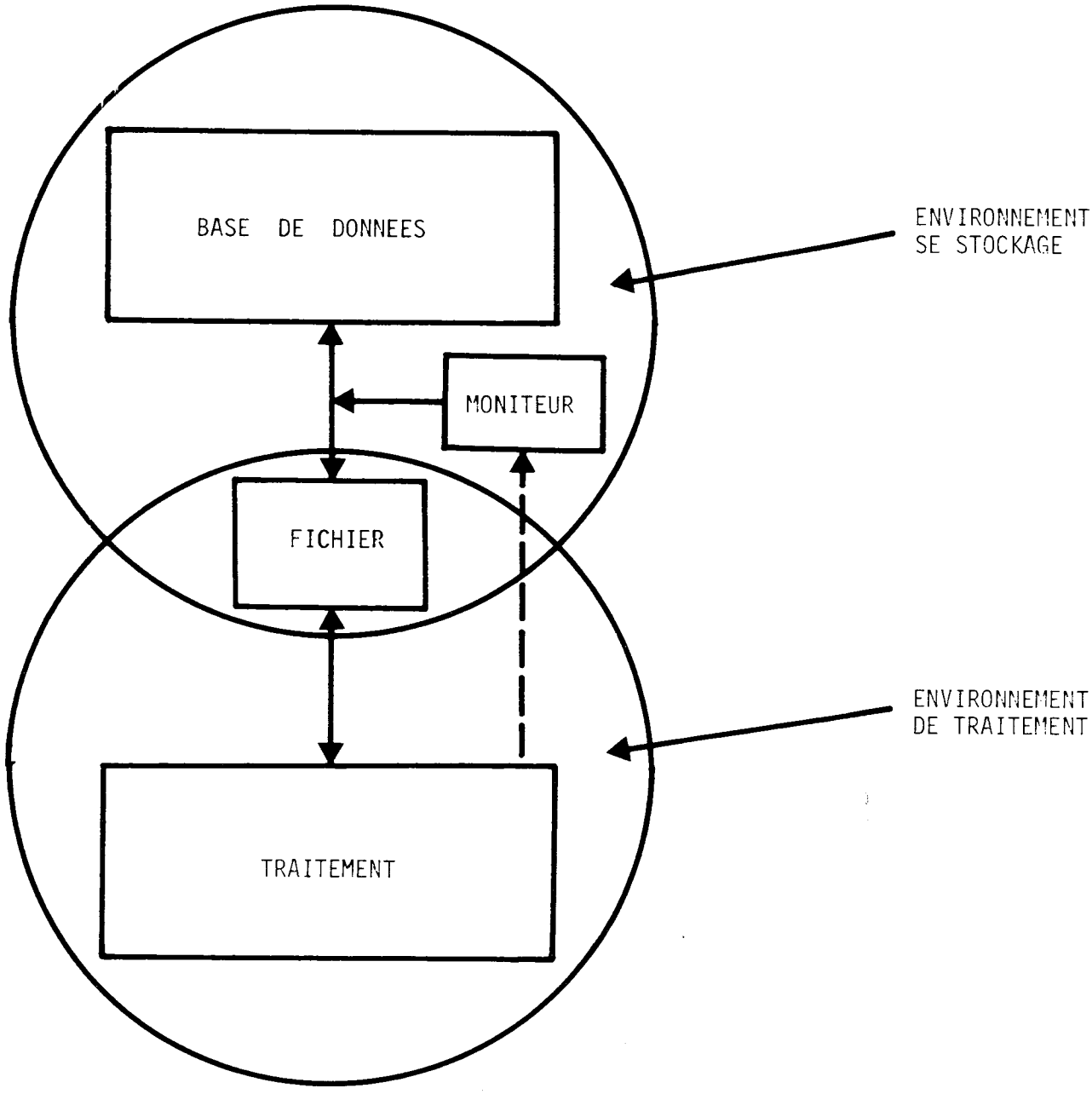


FIG. II-13 Organisation par base de données et fichier

Bibliographie

|1| ALEXANDER Ch.
De la Synthese de la Forme, essai
Dunod / Collection Aspects de l'Urbanisme / 1971

|2| BASKIN H. S.
A comprehensive applications methodology for symbolic computer graphics
in: Pertinent concepts in computer graphics / University of Illinois Press / 1969

|3| BIERSTONE E., BERNHOLTZ A.
HIDECS - RECOMP Procedure
Rapport / Department of Civil engineering / M.I.T.1967

|4| BLAIN G., LABARTHE A., RAULT JC., SCIARDIS M., ZAMANSKY P.
BAL: An aid to scientific programming based upon a "bank of algorithms"
Proc. IFIP 1974

|5| BOLOPION A., CAMPMAS M., LATOMBE JC., SABONNADIERE LC
L'interpreteur-generateur de langages de commandes
Laboratoire d'electrotechnique / INPG / 1973

|6| BRACCHI C., FERRARI D.
A language for treating geometric patterns in a two-dimensional space
CACM / Vol. 14 / No. 1 / pp.26-32 / 1971

|7| BRESSY Y., DAVID B., FANTINO Y., MERMET J.
A hardware Compiler for Interactive Realisation of Logical Systems described in CASSANDRE
Workshop on Computer Hardware Description Languages and Their Applications / New York / sept. 3-5 / 1975

|8| COONS S. A.
An Outline of the Requirements for a Computer-Aided Design
AFIPS Proc. SJCC / Vol. 23 / pp.299-304 / 1963

|9| CRESPI-REGLIZZI S., MORPURGO R.
A language for treating graphs
CACM / Vol. 3 / No. 5 / pp. 319-323 / 1970

|10| CUNIN P.Y., DELAUNAY M., SIMONET M., VOIRON J.
Aspects d'une methodologie d'ecriture de compilateurs
Laboratoire d'informatique / USMG / 1975

|11| DI RUZZA F., LE FAOU C., MERMET J., RECHENMANN F.
Computer Tools for the Construction of Dynamic Space-distributed Simulation Models
Rapport ENSIMAG / 1974

|12| DUCROT A
Principes d'organisation et de realisation du systeme graphique interactif GIPSY
These de 3e cycle / Lille / 1974

|13| Enseignement de la programmation
Semiraire / Laboratoire d'informatique / USMG / 1973-74

|14| EASTMAN C. M., YESSIOS C. I.
An efficient algorithm for finding the union, intersection and differences of spatial domains
Rapport / Departement of Computer Science / Carnegie-Mellon University / 1972

|15| GOSS G., HARTMANIS J./ed./
Compiler Construction
An Advanced Course
Spring-Verlag / 1974

|16| HAMBURY J. N., KARNIEL S.
Modular graphic software for CAD
in ONLINE 72 / 1972

|17| JACQUART R., REGNIER P., VALETTE F. R.
GERMINAL, un systeme integre conversationnel pour la C.A.O.
Automatisme / Tome XIX / No. 3 / pp.163-168 / 1974

|18| JOHNSON W.L., PORTER J.H. ACKLEY I.S., ROSS D.T.
Automatic Generation of efficient Lexical Processors using Finite State Techniques
CACM / Vol.11 / no.12 / pp.805-813 / 1968

|19| JORRAND Ph.
Contribution au developpement des langages extensibles
These d'etat / Grenoble / 1975

|20| KNUDSEN M. J.
PMSL, an interactive language for system-level description and analysis of computer structures
Rapport / Dep. of Computer Science / Carnegie-Mellon University / 1973

|21| LE FAOU C., SICOT J.P.
IMAG III, Systeme de simulation de circuits electroniques
Electronique et microelectronique industrielle / novembre 1974

|22| LUCAS M.
Technologie, programmation et utilisation des consoles de visualisation (etat de recherches actuelles)
Rapport / Laboratoire d'informatique / USMG / 1974

|23| MARCH L., STEADMAN Ph.
The geometry of environment (An introduction to spatial organisation in design)
RIBA Publications Ltd / 1971

|24| MERRILL R. D.
Representation of Contours and Regions for Efficient Computer Search
CACM / Vol. 16 / No. 2 / pp. 69-82 / 1973

|25| MORSE S. P.
Graphical modelling using contextually implied functions
Colloque: les traitements graphiques par ordinateur / Grenoble / Avril 1970

|26| SLEIGHT T. P., KOSSIAKOFF A.
Use of graphics in software design development and documentation
Journees graphiques 1973 / AFCET-IRIA / December 1973

|27| UNGER C./ed./
Command Languages
North-Holland pub.c. 1975

|28| WARD J. E.
Systems engineering problems in computer-driven CRT displays for man-machine communication
IEEE / Vol. SSC-3 / No. 1 / pp. 47-54 / 1967

CHAPITRE III

APPLICATION: CASSANDRE

III.0 Avertissement

Cette application, par laquelle nous avons commencé notre travail nous a servi pour dégager les principaux concepts. Pendant la présentation nous allons la replacer dans le nouveau contexte et elle nous servira à illustrer certains propos tenus dans les chapîtres précedents.

Après une brève présentation de Cassandre nous décrirons un procédé qui, à partir d'une description structurelle et fonctionnelle en produit une autre, équivalente, dans laquelle on sépare ces deux aspects. De cette façon nous obtenons une description qui correspond au MSR (description structurelle) et ceci nous permet de considérer le système développé comme une particularisation du système SIGMA-C.A.O..

Après cela nous présentons le problème de la traduction d'un langage de description dans la structure interne (MSRi) et un traitement de déduction des données topologiques absentes dans la description d'origine et indispensable pour une présentation graphique.

La description des manipulations de schémas produits correspondant à l'utilisation particulière du modèle pour les manipulations symboliques termine ce chapitre.

III.1 Présentation du langage

III.1.1 Présentation du langage et du système Cassandre.

Parmi les langages et systèmes pour la C.A.O. des systèmes logiques on trouve Cassandre (13), (14). De nombreux travaux ont permis d'élaborer un système complet de programmes permettant d'obtenir à partir d'une description en langage telle ou telle application. Citons à titre d'exemple la production de microprogrammes (7), la simulation de fonctionnement du système décrit (11), etc... Le langage Cassandre, proche syntaxiquement d'ALGOL, permet une description structurelle et fonctionnelle d'un système logique. Grâce à sa structure emboîtée, l'utilisateur peut organiser la description de son projet de la façon qui lui paraît la plus appropriée; par la notion d'unité (correspondant à l'entité du chapître I) le langage facilite tant l'écriture que la compréhension. Les points suivants représentent les différentes possibilités d'unités.

-- la plus simple des unités peut être réduite à un fil (une entrée et une sortie).

-- chaque unité peut contenir un réseau d'unités interconnectées.

-- il existe une unité qui contient toutes les autres.

-- une description en Cassandre peut être considérée comme un arbre dont chaque noeud est un réseau d'unités.

Le texte en langage Cassandre décrivant une unité contient principalement deux descriptions, de nature différente (Fig.III-1):

-- Partie structurelle:

-- entête d'unité avec ses entrées-sorties.
-- déclarations propres à l'unité.
-- connections permanentes et inconditionnelles.

-- Partie fonctionnelle:

description des fonctions logiques que cette unité est censée effectuer.

Cassandre possède une gamme d'instructions fonctionnelles importante. Nous citerons les instructions de chargement de registres, de branchements conditionnels, les expressions logiques, ...

II.1.2 Exemples

Les exemples que nous avons choisis pour illustrer ce chapitre sont:

-- un compteur de 4 bits que l'on peut charger, remettre à zero et lui faire plus et moins un

-- un compteur de 8 bits réalisé avec deux compteurs de 4 bits et ayant en plus la possibilité d'effectuer les décalages à gauche

On trouve leurs descriptions en CASSANDRE sur la Fig.III-1.

```
'UNITE' COMP4 (H,P(0:3),PE,CU,CD,RAZ;Q(0:3),TC);
'REGISTRE'  CT(1:4);
'SIGNAL' M01(0:3), M02(0:3);
'HORLOGEMERE' H;

Q := CT;
M01 := *D|1|(CT.M02 + CT.M01 + M01.M02) & 0 ;
'SI' CU & CD 'ALORS' (M02 := 0001; TC := /.CT )
                     (M02 := 1111 ; TC := /.(-CT));
<H> 'SI' PE & (CU + CD) & RAZ 'ALORS'
                     ( CT <= P )
                     (CT <= CT # (M02 # M01))
                     ( CT <= 0000 );


'UNITE' COMP8DEC (H,IN(0:7),PE,CU,CD,RAZ,SH;OUT(0:7),TC);
'SIGNAL'  CD1, CU1, TC1, TC2, PG(0:7), Q(0:7);
'HORLOGEMERE' H;
'EXTERNE' COMP4('HORLOGE',(0:3),,,,;(0:3),);

'SI' SH & PE 'ALORS'  (PG:= *D|1|(Q)&0)
                      (PG:=IN) ;
CU1 := CU . TC2 ;
CD1 := CD . TC2 ;
'SI' CU & CD 'ALORS' (TC:=TC2)
                     (TC:=TC1);
COMP4|1| (H,PG(0:3),PE,CD1,CU1,RAZ;Q(0:3),TC1);
COMP4|2| (H,PG(4:7),PE,CD,CU,RAZ;Q(4:7),TC2);
OUT := Q ;
```

FIG. III-1 Description en CASSANDRE de COMP4 et de COMP8DEC

III.1.3 Compilateur de hardware

Commençons par une remarque:
cette démarche effectuée originellement pour obtenir une description structurelle en vue d'une réalisation physique nous a surtout intéressé pour la raison suivante: elle permet d'obtenir à partir d'une description structurelle et fonctionnelle une description dans laquelle on ne fait intervenir les notions fonctionnelles qu'au niveau des unités primitives.

On veut traduire une description logique en une description plus proche de la réalisation; on dit qu'une unité est realisée si l'on arrive à obtenir sa description structurelle complète. Pour cela il faut non seulement tenir compte de la partie structurelle de la description mais encore remplacer la partie fonctionnelle par un réseau (c'est-à-dire une liste de connections permanentes et inconditionnelles); montrons le sur des exemples:

-- Si1 := Si2 est une instruction inconditionnelle de connection entre les signaux Si1 et Si2 , donc structurelle; pendant la réalisation on ne la touchera pas.

-- l'instruction allera E1 est une instruction purement fonctionnelle indiquant le changement d'état; lors de la réalisation on la place dans une unité fonctionnelle particulière appelée CONTROLE.

-- si CN alors Si := Si1

sinon Si := Si2

est une instruction à la fois structurelle et fonctionnelle.

Lors de la réalisation on la remplacera par:

```
BUS (ET|1| (CN, Si 1 ; *),
     ET|2| (NØN(CN ; *), Si 2 ; *) ; Si)
```

ce qui correspond au schéma suivant:

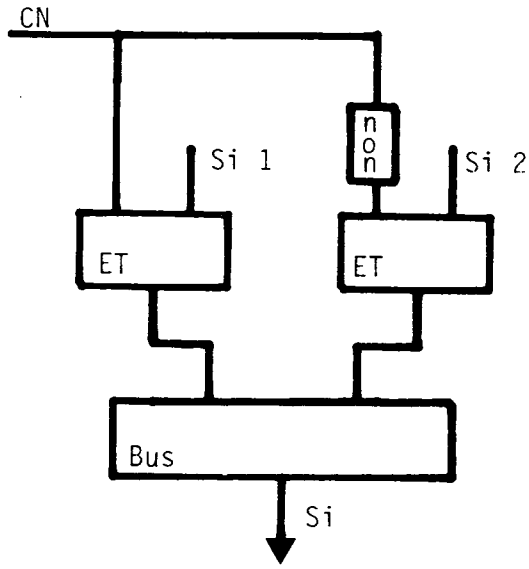


FIG. III-2 Réalisation d'une conncexion conditionnelle

On peut constater que la description obtenue et surtout le schéma sont plus proches de la réalisation en circuits logiques que la description initiale. C'est dans un but de réalisation physique que successivement Liddell (11) et Fantino (5) ont étudié et implémenté ce traitement en l'appelant SURDECOUPAGE ou COMPILATEUR DE HARDWARE: il s'agit d'une transformation des notions fonctionnelles complexes du langage en une structure de composants fonctionnels élémentaires.

Les seules notions du langage qui interviennent dans la description structurelle sont:

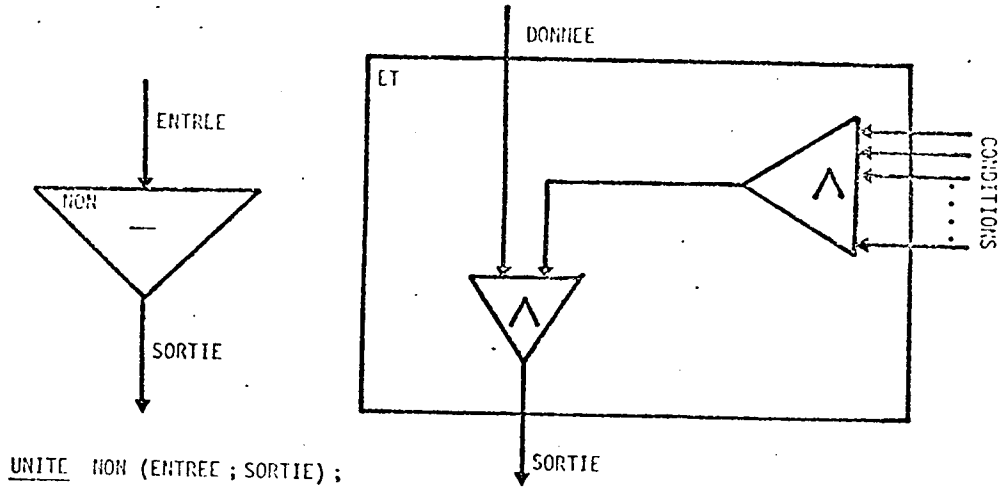
-- l'unité avec son entête.
-- les déclarations de signal, de signaux-unités et de signaux.
-- éventuellement, les notions arithmétiques à fonction descriptive des connections répétitives.

qui sont celles qui permettent la description de réseaux de boîtes interconnectées.

Toute description en Cassandre peut contenir ces notions mais une description qui ne contient qu'elles sera dite faite en Cassandre-E (élémentaire).

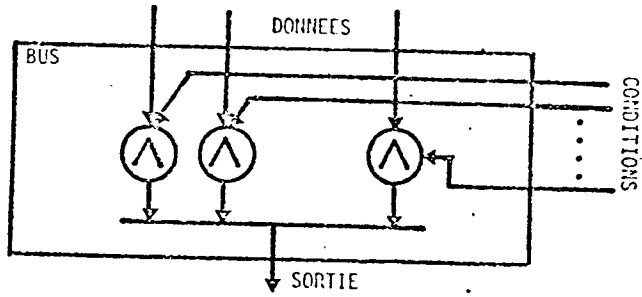
On considère que réaliser une unité c'est passer de sa description en Cassandre en une description logiquement équivalente en Cassandre-E.

Cette transformation est une compilation particulière qui se traduit par un appauvrissement de la syntaxe. Pour l'effectuer il faut associer à toute notion syntaxique non contenue dans Cassandre-E, un schéma d'expression en Cassandre-E. Le choix de l'ensemble d'unités primitives pouvant traduire toutes les notions fonctionnelles de Cassandre est primordial. Nous donnons sur la Fig III-3 les unités primitives choisies par Fantino dans le compilateur de hardware et sur la Fig. III-4 les descriptions transformées des deux compteurs.



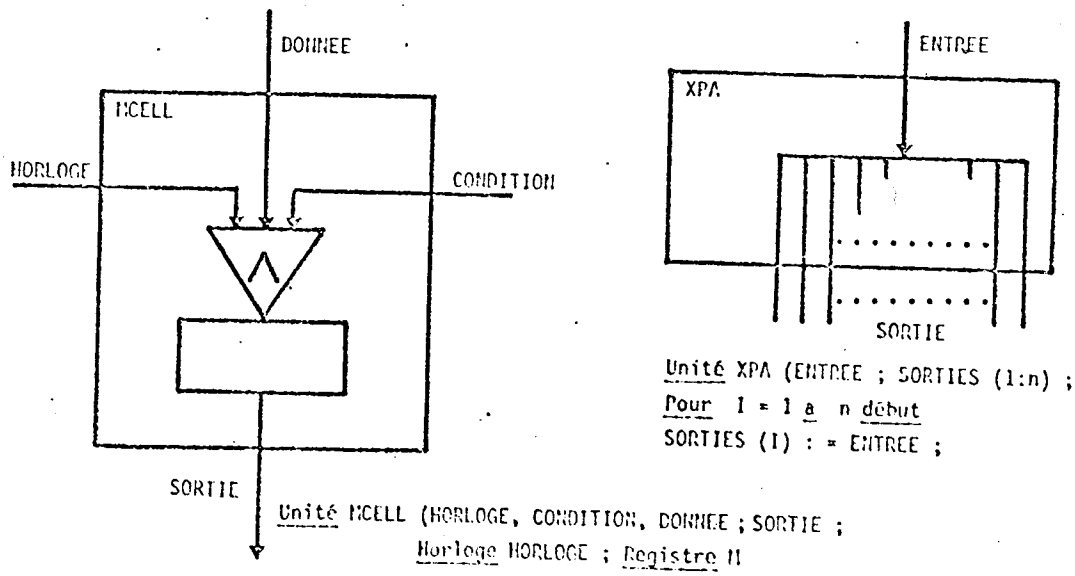
UNITE NON (ENTREE ; SORTIE) ;
SORTIE : = ENTREE ;

UNITE ET (DONNEE, CONDITIONS (1:n) ; SORTIE) ;
SORTIE : = DONNEE ∧ (/∧ CONDITIONS) ;



UNITE BUS
CONDITIONS (1:n),
DONNEES (1:n) ;
SORTIES) ;

SORTIE : = si
CONDITIONS alors
DONNEES ;



Unité MCELL (HORLOGE, CONDITION, DONNEE ; SORTIE ;
Horloge HORLOGE ; Registre M
<HORLOGE> si CONDITION alors M : = DONNEE ;
SORTIE : = M ;

Unité XPA (ENTREE ; SORTIES (1:n) ;
Pour I = 1 à n début
SORTIES (I) : = ENTREE ;

FIG. III-3 Unités de base avec leurs descriptions en CASSANDRE

```
'UNITE' EXP00000 (E000(0:3),E001;S (0:3));
          S:=*D|1| E000 & E001 ;
'UNITE' EXPC0001 (E000(0:3),E001(0:3),E002(0:3),E003(0:3),E004(0:3),
E005(0:3);S (0:3));
          S:= E000 . E001 + E002 . E003 + E004 . E005 ;
'UNITE' EXP00002 (E000,E001;S (0:1));
          S:= E000 & E001 ;
'UNITE' AND00000 (COND,E (0:3);S (0:3));
  'EXTERNE' AND(,;);
           POUR' I=0 'A' 3
        'DEBUT'
           AND|||(COND,E(I);S(I));
        'FIN';
'UNITE' EXP00003 (E000(0:3);S );
          S:=/. E000 ;
'UNITE' EXP00004 (E000(0:3);S (0:3));
          S:=- E000 ;
'UNITE' EXP00005 (E000,E001,E002;S (0:2));
          S:= E000 & E001 & E002 ;
'UNITE' EXP00006 (E000,E001;S );
DEV  180 ATTACHED
          S:= E000 + E001 ;
'UNITE' EXP00007 (E000(0:3),E001(0:3);S (0:3));
          S:= E000 # E001 ;
'UNITE' REG00000 (H,ENABLE(0:3),INFO(0:3);OUTPUT(0:3));
  'HORLOGEMERE' H;
  'EXTERNE' MCELL('HORLOGE',,;);
          'POUR' I=0 'A' 3
        'DEBUT'
           MCELL |||  (H,ENABLE(I),INFO(I);OUTPUT(I));
        'FIN';
'UNITE' XPA00000(IN;OUT(0:3));
          'POUR' I=0 'A' 3
        'DEBUT'
           OUT(I):=IN;
        'FIN';
'UNITE' BUS00000(E000(0:3),E001(0:3);S (0:3));
          S:=E000+E001;
'UNITE' BUS00001(E000(1:1),E001(1:1);S (1:1));
          S:=E000+E001;
'UNITE' BUS00002(E000(1:4),E001(1:4),E002(1:4);S (1:4));
          S:=E000+E001+E002;
'UNITE' BUS00003(E000(1:4),E001(1:4),E002(1:4);S (1:4));
          S:=E000+E001+E002;
```

FIG. III-4-a Description de COMP4 en CASSANDRE-E

```
'UNITE' COMP4(H,P(0:3),PE,CU,CD,RAZ;Q(0:3),TC);
  'SIGNAL' CT(1:4);
  'SIGNAL' MO2(0:3),MO1(0:3);
  'SIGNAL' SGL00001(1:4),SGL00000(1:4),CND00001(0:2),CND00000(0:1);
  'HORLOGEMERE' H;
  'EXTERNE' EXP00000 ((0:3),;(0:3)),EXP00001 ((0:3),(0:3),(0:3),(0:3),
  (0:3),(0:3);(0:3)),EXP00002 (,;(0:1)),EXP00003 ((0:3);),EXP00004 ((0
  :3);(0:3)),EXP00005 (,,;(0:2)),EXP00006 (,;),EXP00007 ((0:3),(0:3);(
  0:3));
  'EXTERNE' AND(,;),AND00000(,(0:3);(0:3));
  'EXTERNE' XPA00000(;(0:3));
  'EXTERNE' REG00000('HORLOGE',(0:3),(0:3);(0:3));
  'EXTERNE'BUS00000((0:3),(0:3);(0:3)),BUS00001((1:1),(1:1);(1:1)),
  BUS00002((1:4),(1:4),(1:4);(1:4)),BUS00003((1:4),(1:4),(1:4);(1:4));
          Q(0:3):=CT(1:4);
          EXP00000|0|(EXP00001|0|(CT(1:4),MO2(0:3),CT(1:4),MO1(0:3),
          MO1(0:3),MO2(0:3);*),0;MO1(0:3));
          EXP00002|0|(CU(1),CD(1);CND00000(0:1));
          EXP00005|0|(PE(1),EXP00006|0|(CU(1),CD(1);*),RAZ(1);CND00001
          (0:2));
          REG00000|0|(H(1),SGL00001(1:4),SGL00000(1:4);CT(1:4));
          BUS00000(AND00000|0|(CND00000(0),0001;*),AND00000|1|(
          CND00000(1),1111;*);MO2(0:3));
          BUS00001(AND|2|(CND00000(0),EXP00003|0|(CT(1:4);*);*),AND|3|
          (CND00000(1),EXP00003|1|(EXP00004|0|(CT(1:4);*);*);*);TC(1))
          ;
          BUS00002(AND00000|2|(CND00001(0),P(0:3);*),AND00000|3|(
          CND00001(1),EXP00007|0|(CT(1:4),EXP00007|1|(MO2(0:3),MO1(0:3
          );*);*);*),AND00000|4|(CND00001(2),0000;*);SGL00000(1:4));
          BUS00003(XPA00000|0|(CND00001(0);*),XPA00000|1|(CND00001(1);
          *),XPA00000|2|(CND00001(2);*);SGL00001(1:4));
```

FIG. III-4-a Description de COMP4 en CASSANDRE-E (suite)

```
'UNITE' EXP00000 (E000,E001;S (0:1));
          S:= E000 & E001 ;
'UNITE' EXP00001 (E000(0:7),E001;S (0:7));
          S:=*D|1| E000 & E001 ;
'UNITE' AND00000 (COND,E (0:7);S (0:7));
  'EXTERNE' AND(,;);
          'POUR' I=0 'A' 7
        'DEBUT'
            AND|I|(COND,E(I);S(I));
        'FIN';
'UNITE' EXP00002 (E000,E001;S );
          S:= E000 . E001 ;
'UNITE' BUS00000(E000(0:7),E001(0:7);S (0:7));
          S:=E000+E001;
'UNITE' BUS00001(E000(1:1),E001(1:1);S (1:1));
          S:=E000+E001;
'UNITE' BUS00002(E000(0:3),E001(4:7);S (0:7));
          S:=E000&E001;
'UNITE' COMP8DEC(H,IN(0:7),PE,CU,CD,RAZ,SH;OUT(0:7),TC);
  'SIGNAL' Q(0:7),PG(0:7),TC2,TC1,CU1,CD1;
  'SIGNAL' CND00001(0:1),CND00000(0:1);
  'HORLOGEMERE' H;
  'EXTERNE' COMP4('HORLOGE' ,(0:3),,,,;(0:3),);
  'EXTERNE' EXP00000 (,;(0:1)),EXP00001 ((0:7),;(0:7)),EXP00002 (,;);
  'EXTERNE' AND(,;),AND00000(,(0:7);(0:7));
  'EXTERNE'BUS00000((0:7),(0:7);(0:7)),BUS00001((1:1),(1:1);(1:1)),
  BUS00002((0:3),(4:7);(0:7));
          OUT(0:7):=Q(0:7);
          EXP00002|0|(CU(1),TC2(1);CU1(1));
          EXP00002|1|(CD(1),TC2(1);CD1(1));
          COMP4|1|(H(1),PG(0:3),PE(1),CD1(1),CU1(1),RAZ(1);,TC1(1));
          COMP4|2|(H(1),PG(4:7),PE(1),CD(1),CU(1),RAZ(1);,TC2(1));
          EXP00000|0|(SH(1),PE(1);CND00000(0:1));
          EXP00000|1|(CU(1),CD(1);CND00001(0:1));
          BUS00000(AND00000|0|(CND00000(0),EXP00001|0|(Q(0:7),0;*);*),
          AND00000|1|(CND00000(1),IN(0:7);*);PG(0:7));
          BUS00001(AND|2|(CND00001(0),TC2(1);*),AND|3|(CND00001(1),TC1
          (1);*);TC(1));
          BUS00002(COMP4|1|(,,,,,;*,),COMP4|2|(,,,,,;*,);Q(0:7));
```

FIG. III-4-b Description de COMP8DEC en CASSANDRE-E

III.1.4 Langage de description et MSR

En ce qui concerne la relation entre Cassandre-E et les structures du formalisme MSR nous constatons que:

-- Cassandre-E ne peut représenter que des graphes orientés.

-- une seule relation est utilisée qui est la relation de connection ayant comme donnée quantitative la dimension de la voie ce qui réduit le MSR à un sur-réseau (SR).

-- les unités n'ont pas de propriétés propres ce qui facilite l'implémentation des opérations de restructuration.

On peut donc représenter un programme écrit en Cassandre-E à l'aide d'une structure MSR.

Remarques:

-- il aurait été intéressant d'avoir deux types de relations pour pouvoir distinguer facilement les chemins de données et les chemins de contrôle.

-- la présence des propriétés propres à l'unité permettrait une implémentation facile de fonctions d'évaluation: par exemple le nombre d'unités contenues dans l'unité englobante.

III.2 Production automatique de schéma

Nous avons pu dissocier la production automatique de schémas pour les raisons suivantes:

-- nature non conversationnelle des traitements utilisés.

-- l'utilisateur ne veut produire les schémas que dans un but de documentation et n'a donc besoin ni de terminal graphique ni de possibilités conversationnelles.

On peut donc l'utiliser avec des moyens réduits.

Pour obtenir, à partir de la description en langage, le schéma logique correspondant, différentes étapes sont nécessaires (Fig. III-5). Tout d'abord il faut extraire du langage les informations dont on aura besoin; puis leur adjoindre des données topologiques induites. Il faut enfin effectuer une répartition dans le plan pour que les connections que l'on tracera puissent mettre en évidence les relations entre unités.

Nous allons exposer maintenant les traitements effectués pour chacune des étapes.

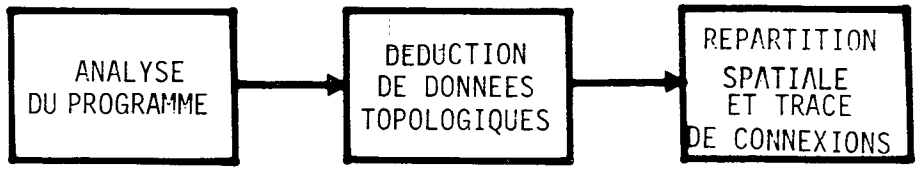


FIG. III-5 Passage du langage au schéma

III.2.1 Analyse du programme

On procède en partant de l'unité principale, puis on examine toutes les unités non primitives. On ne s'intéresse pas au contenu des unités primitives et on les note en tant que telles car elles contiennent une description fonctionnelle que nous ignorons lors de l'élaboration du schéma. Pour que cette distinction puisse être faite il faut que l'analyseur reconnaisse ces unités primitives. Des unités primitives standards (dont le compilateur se servira lorsqu'il générera des unités) sont prédéclarées dans l'analyseur: Si l'utilisateur désire les changer ou en ajouter d'autres il faut les spécifier lors de l'initialisation de l'analyseur.

Pour une unité il faut obtenir trois types d'informations lors de l'analyse du programme:

-- tous les objets (nommés) s'y trouvant.

-- toutes les unités contenues (afin de pouvoir construire l'arbre d'appel).

-- les relations entre les objets de l'unité.

Nous obtenons les deux premiers types d'information en analysant les déclarations, le troisième type se trouve dans la partie instruction et nous la traitons à l'aide d'un analyseur syntaxique.

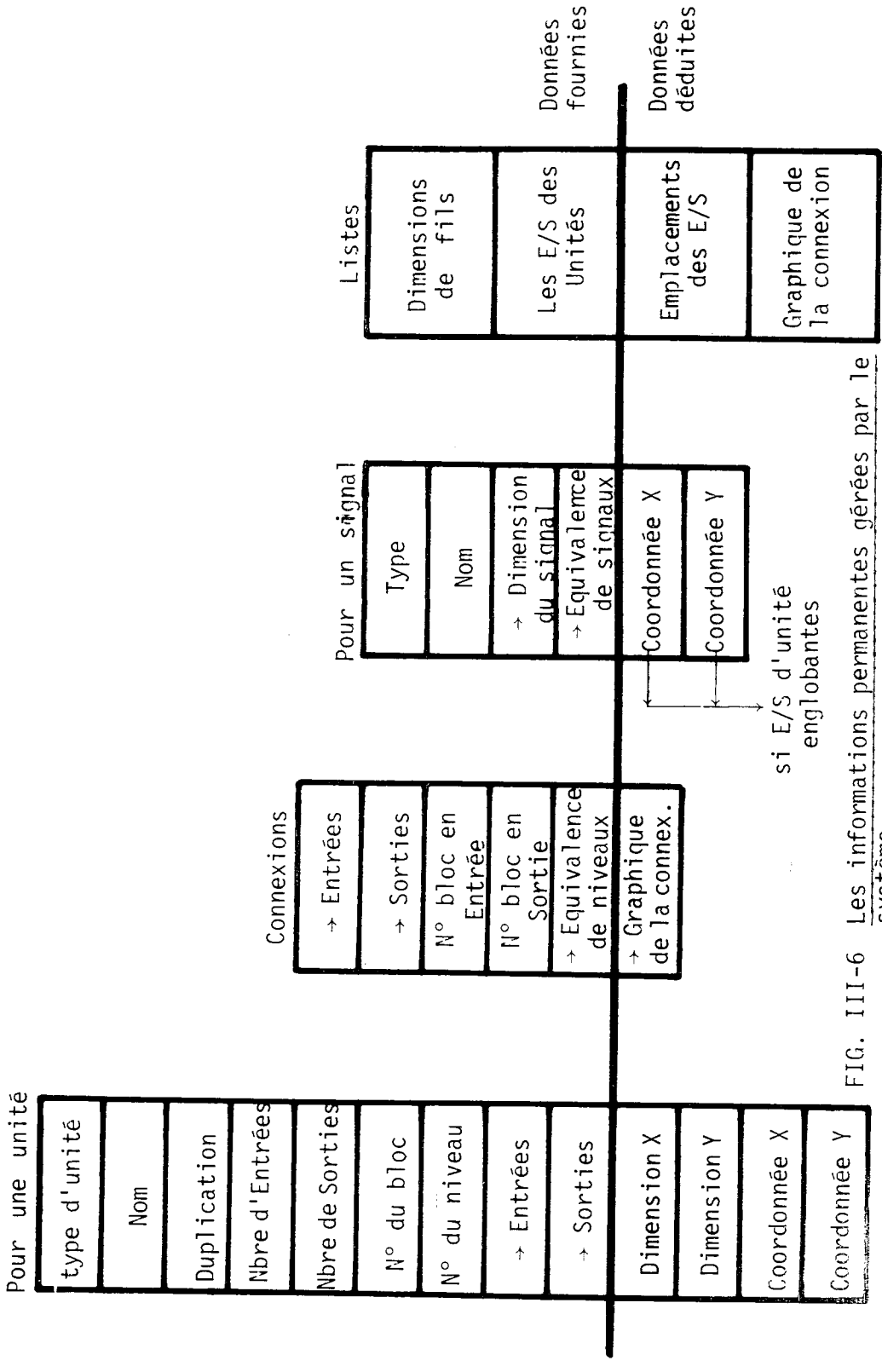


FIG. III-6 Les informations permanentes gérées par le système

Remarque : Pour minimiser la durée des traitements nous n'analysons chaque unité qu'une seule fois même si elle est utilisée, grâce aux duplications, plusieurs fois sur le même niveau ou même sur des niveaux différents. Cela a pour conséquence que l'arbre obtenu à la fin du processus d'extraction n'est pas complet mais contient toutes les informations vitales pour le devenir (en ne travaillant que sur lui).

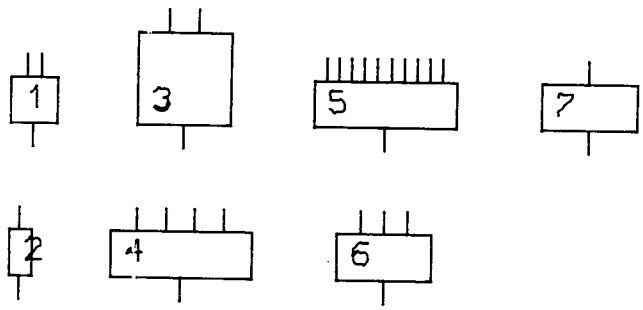
A la fin de cette phase nous abandonnons la description sous forme de langage sachant que nous avons dans la description interne tous les renseignements nécessaires.

Sur la figure III-6 nous montrons les informations permanentes stockées dans le système et leur mode d'obtention. (Les informations propres à chaque traitements ainsi que l'organisation exacte des données ne sont pas présentés).

III.2.2 Association de données topologiques

Etant donné que le langage ne contient aucune spécification topologique il faut, pour pouvoir construire une représentation graphique, interpréter les renseignements logiques disponibles et leur faire correspondre des données topologiques; par exemple le nombre d'unités présentes et leurs types nous permet de déterminer l'unité de mesure (nécessaire à la visualisation) qui intervient dans la détermination de la forme et de la taille de chaque unité interne, en

tenant compte de facteurs tels que le nombre d'entrées et de sorties et les formes qui les caractérisent(fig.III-7).



1 == AND
2 == NON
3 == EXPR
4 == BUS1
5 == BUS2
6 == MCELL
7 == XPA

FIG. III-7 Les schémas d'unités de base

Compte tenu de l'objectif que nous nous sommes fixés, c'est à dire fournir un support pour une documentation automatique et en aucun cas un support pour une quelconque réalisation physique, nous rappelons ici la forme de représentation choisie:

-- le schéma doit présenter sous une forme compréhensible ce que le langage permet de décrire.

-- conformément à la possibilité de description hiérarchique le schéma présente un ou au maximum deux niveaux à la fois.

-- le schéma ne traduit que la description correspondant à la description en Cassandre-E.

-- les unités sont représentées par des rectangles avec les entrées en haut et les sorties en bas.

-- les unités primitives sont reconnaissables par des formes caractéristiques.

-- le langage permet de grouper les fils en paquets logiques. La finesse de représentation des connections est telle qu'elle ne permet de ne prendre en compte que ce découpage logique: on ne peut travailler au niveau des fils sous peine de rendre le schéma trop complexe et moins compréhensible.

Exemple: unité UAL (1:3, (1:3, 1:2), 1, 1:5; 1:2, 1, (1:3, 1:3));
ce qui est avant le point virgule décrit les entrées, ce qui est après les sorties. Le paquet d'entrées est découpé en quatre parties logiques, le paquet de sorties en trois parties; chacune des parties peut être constituée d'un paquet de fils. Pour la visualisation nous nous placerons au niveau du découpage logique et nous allons donc représenter chaque entrée/sortie par un seul fil. (voir fig.III-8).

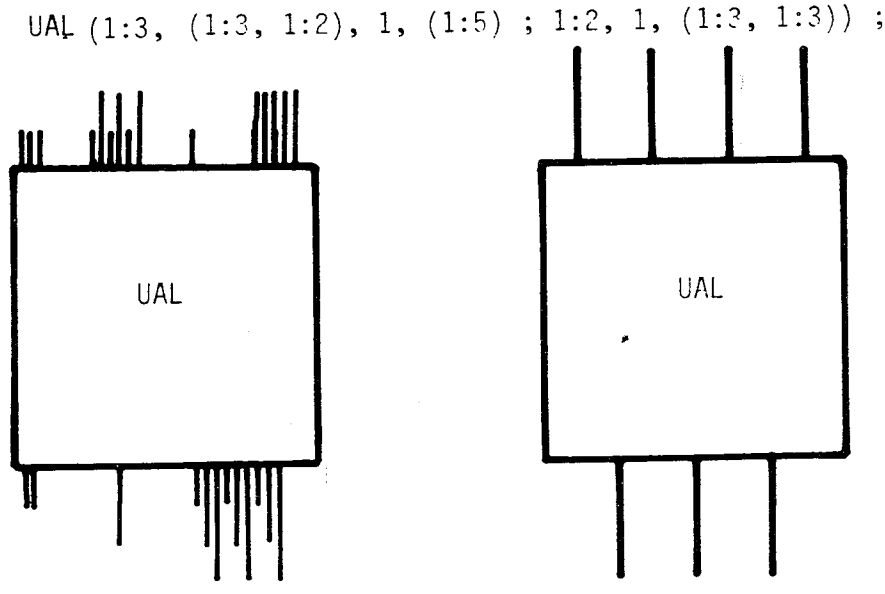


FIG. III-8 Niveau de représentation choisi

-- les connections sont tracées par des segments horizontaux ou verticaux avec les restrictions suivantes:

  -- les connections ne peuvent traverser les unités.
  -- les connections ne peuvent se chevaucher. Par contre aucune limitation n'est imposée quand à leurs croisements.

III.2.3 Allocation spatiale

Après avoir déterminé la forme de toutes les unités nous allons procéder à leur répartition dans le plan en tenant compte de leurs relations. Ce problème appelé allocation spatiale sera traité plus en détail dans le chapître IV.

Nous présentons ici une méthode travaillant en deux étapes que nous avons développée pour montrer que, si on dispose de renseignements précis, on peut résoudre l'allocation spatiale assez facilement par un procédé algorithmique.

Une remarque s'impose quant à la méthode utilisée par Saillard (13) dans DESMAG. Cette méthode résoud d'une façon mathématique le problème de la répartition, en considérant des objets à placer réduits à un point, et les place dans les barycentres. Cela nécessite une étape d'amélioration (suppression de chevauchements) même pour des exemples très simples.

III.2.3 Pour une généralisation des systèmes C.A.O.

En ne considérant plus les éléments en tant que points mais comme des surfaces, on ne peut plus utiliser des résolutions analytiques et on se dirige vers des résolutions combinatoires avec des heuristiques réduisant le nombre de tentatives. En contre partie on peut obtenir directement des répartitions valables. Pour les cas difficiles ou très particuliers nous gardons de toute façon la possibilité d'amelioration manuelle.

III.2.3.1 Présentation de la méthode

Rappelons le but de l'operation:

Placer des unités d'une certaine taille dans le plan représentant l'unité englobante, en vue d'effectuer une répartition convenable, en tenant compte de l'unique relation entre ces dernières qui est la relation de connection et sachant que les entrées se trouvent en haut et les sorties en bas du rectangle.

En analysant des programmes générés par le compilateur de hardware (voir fig.III-9), nous avons remarqué plusieurs faits significatifs, en particulier en ce qui concerne les relations entre les unités. Les unités ne sont pas uniformément reliées, mais on trouve toujours des groupements, ce qui n'a rien de surprennant; par contre en analysant ces groupements nous nous sommes aperçus que dans la plupart des cas le groupement prend la forme d'arbre et que seules la racine et les feuilles de ce dernier sont connectées avec

l'extérieur. De plus, ceci étant dû au traitement effectué par le compilateur de hardware un tel groupement est représenté très souvent par une seule instruction, souvent assez complexe, du programme généré.

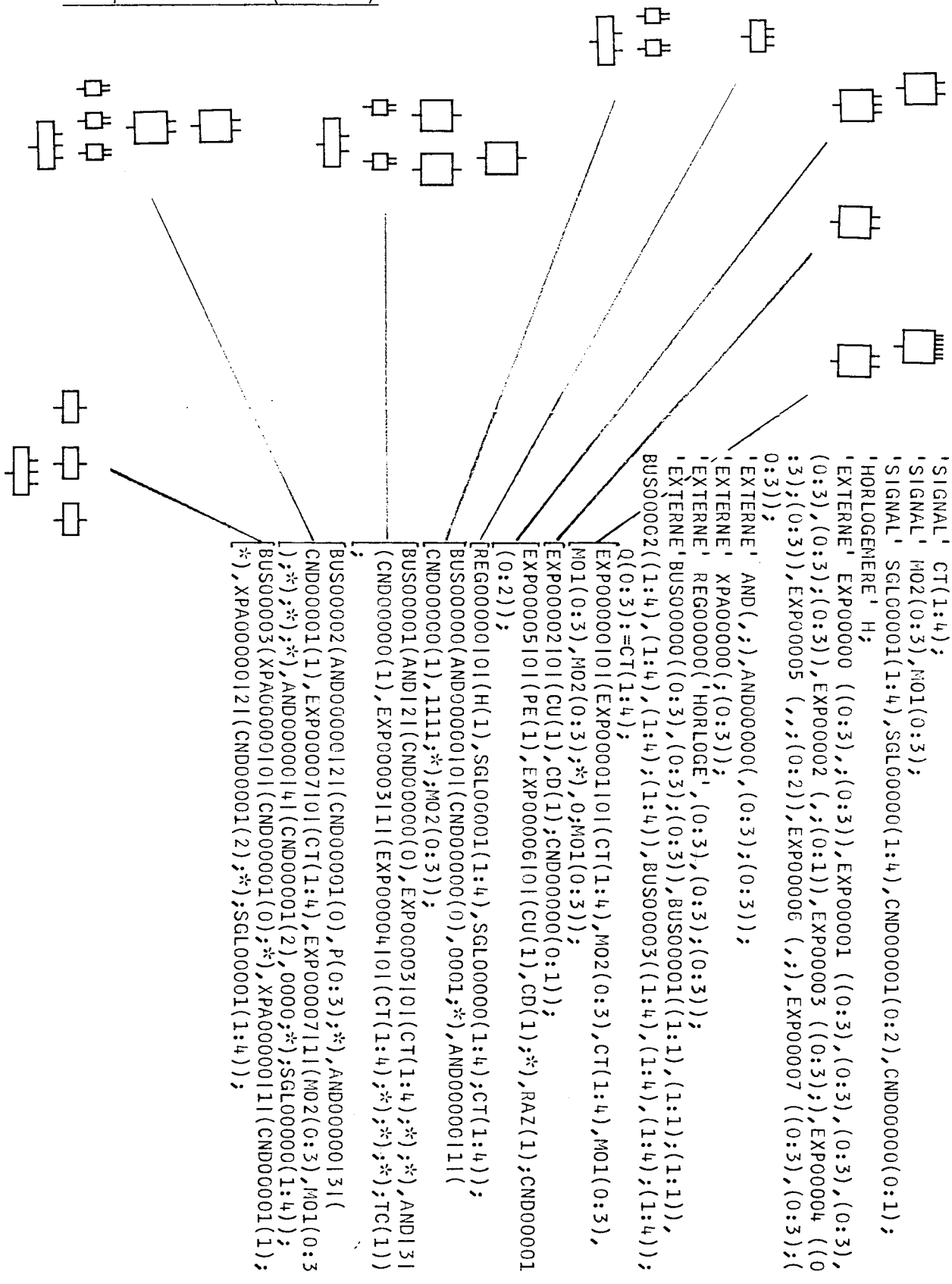
Ces particularités nous ont conduit à choisir les hypothèses suivantes.

Introduire une nouvelle notion, le bloc (voir Fig.III-9), qui va contenir des groupements et aura comme entrées (resp. sorties) des entrées (resp. sorties) reliant des unités contenues dans le bloc avec des unités extérieures à celui-ci.

On effectue le placement en deux phases: d'abord les unités dans les blocs, puis les blocs eux-mêmes: Sachant que le programme source analysé sera toujours généré, soit par le compilateur de hardware, soit par notre programme de sortie, nous allons, compte tenu de la constatation de similitude entre instruction du langage et bloc, faire correspondre ces deux notions. Une conséquence est que l'unité se trouvant dans une instruction numéro N appartiendra au bloc N; au cas, très rare,ou une unité apparaîtrait dans plusieurs instructions nous la mettrons dans le bloc avec lequel elle a le plus de connections.

Ces deux étapes du placement que nous présenterons maintenant sont des méthodes constructives, élément par élément.

FIG. III-10 Exemple de blocs (dessins)



```
'SIGNAL' CT(1:4);
'SIGNAL' MO2(0:3),MO1(0:3);
'SIGNAL' SGLOOOO1(1:4),SGLOOOOO(1:4),CNDOOOO1(0:2),CNDOOOOO(0:1);
'HORLOGEMERE' H;
'EXTERNE' EXPOOOOO ((0:3),;(0:3)),EXPOOOO1 ((0:3),(0:3),(0:3),(0:3),
(0:3),(0:3);(0:3)),EXPOOOO2 (,;(0:1)),EXPOOOO3 ((0:3);),EXPOOOO4 ((0
:3);(0:3)),EXPOOOO5 (,;(0:2)),EXPOOOO6 (,;),EXPOOOO7 ((0:3),(0:3);(
0:3));
'EXTERNE' AND(,;),ANDOOOOO(,(0:3);(0:3));
'EXTERNE' XPAOOOOO(;(0:3));
'EXTERNE' REGOOOOO('HORLOGE',(0:3),(0:3);(0:3));
'EXTERNE'BUSOOOOO((0:3),(0:3);(0:3)),BUSOOOO1((1:1),(1:1);(1:1)),
BUSOOOO2((1:4),(1:4),(1:4);(1:4)),BUSOOOO3((1:4),(1:4),(1:4);(1:4));
    Q(0:3):=CT(1:4);
EXPOOOOO1O1(EXPOOOO1O1(CT(1:4),MO2(0:3),CT(1:4),MO1(0:3),
MO1(0:3),MO2(0:3);*),0;MO1(0:3));
EXPOOOO2O1O1(CU(1),CD(1);CNDOOOOO(0:1));
EXPOOOO5O1O1(PE(1),EXPOOOO6O1(CU(1),CD(1);*),RAZ(1);CNDOOOO1
(0:2));
REGOOOOO1O1(H(1),SGLOOOO1(1:4),SGLOOOOO(1:4);CT(1:4));
BUSOOOOOO(ANDOOOOO1O1(CNDOOOOO(0),0001;*),ANDOOOOO111(
CNDOOOOO(1),1111;*);MO2(0:3));
BUSOOOO1(AND121(CNDOOOOO(0),EXPOOOO3101(CT(1:4);*),AND131
(CNDOOOOO(1),EXPOOOO3111(EXPOOOO4101(CT(1:4);*);*);*);TC(1))
;
BUSOOOO2(ANDOOOOC121(CNDOOOO1(0),P(0:3);*),ANDOOOOO131(
CNDOOOO1(1),EXPOOOO7101(CT(1:4),EXPOOOO7111(MO2(0:3),MO1(0:3
);*);*);*),ANDOOOOO141(CNDOOOO1(2),0000;*);SGLOOOOO(1:4));
BUSOOOO3(XPAOOOOO1O1(CNDOOOO1(0);*),XPAOOOOO111(CNDOOOO1(1);
*),XPAOOOOO121(CNDOOOO1(2);*);SGLOOOO1(1:4));
```

FIG. III-9 Exemple de blocs (en langage)

Lors de l'analyse du programme par l'analyseur syntaxique on affecte à chaque unité un numéro de bloc auquel elle va appartenir et un numéro de niveau dans ce bloc qui correspond au niveau courant d'emboîtement dans l'instruction.

III.2.3.2 Placement à l'intérieur du bloc

Comme les renseignements dont nous disposons sont assez précis, le méthode utilisée sera rapide et précise.

Le nombre de niveaux détermine la hauteur du bloc, la largeur est donnée par le niveau demandant le plus de place (que l'on appelle niveau saturé). C'est lui que l'on place en premier. Après cela on place, en remontant ou en descendant sur les autres niveaux, des unités en appliquant une dispersion barycentrique par rapport aux unités déjà placées sur les autres niveaux et ayant une relation avec l'unité en question.

Après avoir terminé le placement sur un niveau on vérifie s'il n'y a pas de chevauchement (cf. fig.III-10); le cas échéant on arrive à une répartition correcte en écartant les unités se gênant.

III.2.3.3 Placement de blocs

La méthode que nous décrirons ici rappellera plus des méthodes générales que l'on trouve dans la bibliographie.

Nous définissons des fonctions de choix dont nous avons besoin pour déterminer l'ordre de placement des blocs. Elles choisissent le bloc à placer en fonction de liens avec les blocs déjà placés et les blocs non placés.

Pour trouver l'endroit où le bloc choisi sera placé, on détermine l'emplacement idéal pour ces entrées, puis pour ces sorties et on place le bloc le plus près de ces deux endroits. Après le placement de tous les blocs on vérifie le non chevauchement. Dans le cas d'un recouvrement on essaie d'améliorer la répatition en bougeant les blocs mais en respectant leurs positions mutuelles. Si la tentative n'aboutit pas on laisse à l'utilisateur la choix entre un placement avec chevauchement (pouvant être amélioré manuellement par la suite) et l'application au bloc d'un facteur de réduction de deux en vue d'une nouvelle recherche de la place libre.

Après avoir déterminé l'emplacement de tous les blocs il faut représenter les relations liant les unités.

1 == EXP00000|0|
2 == EXP00001|0|
3 == EXP00002|0|
4 == EXP00003|1|
5 == EXP00003|0|
6 == EXP00004|0|
7 == EXP00005|0|
8 == EXP00006|0|
9 == EXP00007|1|
10 == EXP00007|0|
11 == AND|3|
12 == AND|2|
13 == AND00000|4|
14 == AND00000|3|
15 == AND00000|2|
16 == AND00000|1|
17 == AND00000|0|
18 == XPA00000|2|
19 == XPA00000|1|
20 == XPA00000|0|
21 == REG00000|0|
22 == BUS00000
23 == BUS00001
24 == BUS00002
25 == BUS00003

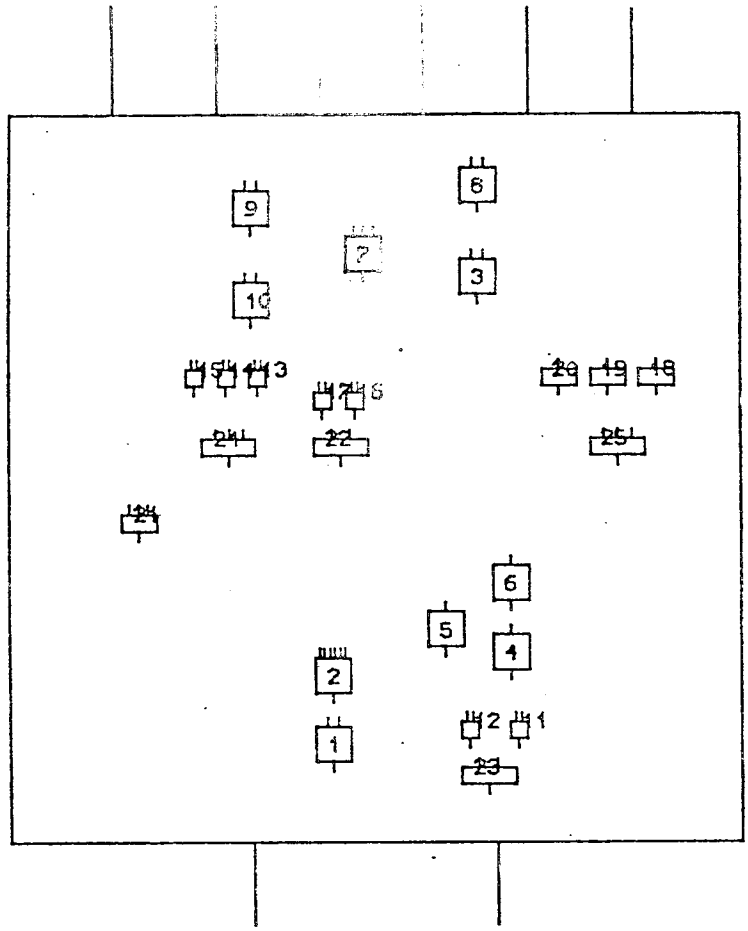


FIG. III-11 COMP4
Après le placement

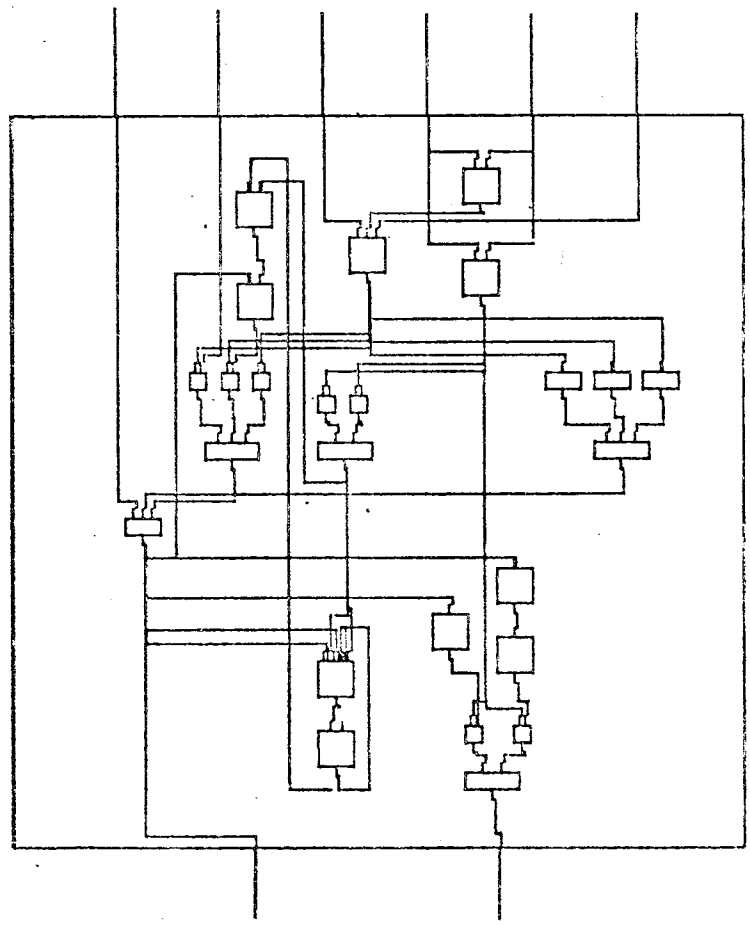


FIG. III-12 COMP4
Schéma complet

III.2.4 Tracé de connections

Dans le langage on exprime les relations de deux façons:

-- soit par ce que l'on appelle connection d'unité; ce qui s'exprime sous la forme: A(,,B(,;*);,) : on ne donne pas de nom et on n'explicite pas la dimension.

-- soit en passant par un signal déclaré; dans ce cas la connection peut prendre des dimensions quelconques, et la relation n'est plus explicite. En effet il faut au moins deux instructions pour la réaliser; dans l'une on précise le départ du signal, dans l'autre l'arrivée. De plus il est tout à fait possible, pour des raisons de commodité, de passer par plusieurs signaux intermédiaires.

Pour pouvoir aborder le tracé, il faut ramener le deuxième cas au premier en retrouvant le ou les signaux caractérisant cette connection.

Ceci peut s'effectuer en deux étapes. D'abord on détermine parmi les connections entre signaux, les classes d'équivalence des signaux. Puis en analysant la matrice contenant les connections UNITE/SIGNAL et SIGNAL/UNITE on détermine le chemin UNITE1/Si1, Si2/UNITE2 avec Si1 équivalent à Si2 et on le remplace par une connection UNITE1/UNITE2 en précisant les signaux équivalents Si1, Si2 la caractérisant.

Après cela le classement de la matrice permet d'obtenir des équipotentielles.

Nous disposons alors de données unifiées et cohérentes pour le tracé des connections. En ce qui concerne l'algorithme de tracé, nous n'entrerons pas dans les détails étant donné que nous n'avons pas utilisé une solution originale et que le nombre d'algorithmes disponibles est assez important. Nous rappelons simplement que nous traçons des segments horizontaux et verticaux en limitant le nombre de changement de direction, en ne se permettant pas la traversée d'unités et le chevauchement de connections déjà existantes. Par contre aucune restriction n'a été donnée quant au nombre de croisements des connections.

Un fois l'analyse du langage source et les traitements indispensables à l'obtention du schéma logique terminés, l'utilisateur peut se contenter du schéma et en demander une sortie sur table traçante, ou l'améliorer en utilisant le système conversationnel.

III.3 Fonctionnement interactif.

Lors de la description des différents stades par lesquels on passe dans le système nous montrons les possibilités qu'il offre.

Si le choix de l'utilisateur se porte vers une utilisation interactive, il précise dans l'environnement d'entrée ce choix et il spécifie le type de travail qu'il veut effectuer (travail nouveau sans description initiale, reprise d'untravail déjà en machine ou un travail sur une description sous forme de langage). Les traitements d'initialisation approprié étant terminés il a à sa disposition les quatre environnements suivants:

-- cataloque
-- modélisation
-- analyse
-- sortie

III.3.1 Environnement de catalogue

Avec les outils de l'environnement de modelisation que nous décrirons par la suite l'utilisateur a la possibilite d'éditer, de modifier et de créer le schéma, l'opération de création ne pouvant utiliser que les unités existantes. Pour pouvoir faire intervenir des unités primitives nouvelles il faut les définir dans l'environnement CATALOGUE.

Quand l'utilisateur désire définir une nouvelle unité primitive il doit passer dans l'environnement de cataloque; cet environnement peut également servir à modifier le fonctionnement d'une unité primitive déjà existante, à condition qu'elle ne soit pas utilisée dans un projet.

Comme déjà signalé les unités primitives sont décrites en Cassandre général; nous sommes donc obligés de les définir d'une façon textuelle et c'est pourquci nous offrons à l'utilisateur les possibilités standards d'édition et de vérification de programmes du système Cassandre.

Après avoir obtenu un programme correct il doit indiquer qu'il s'agit d'une unité primitive et il donne son schéma de représentation. Lorsqu'il revient dans l'environnement de modélisation il peut l'utiliser comme les autres unités.

III.3.2 Environnement de modélisation

On a classé les fonctions de modélisation en cinq familles.

III.3.2.1 Positionnement (Fig.III-13)

La premiere famille de fonctions permet de choisir l'unité que l'on veut étudier. A l'aide de la commande AFFICHER /NOM,PO/, on peut

afficher l'unité que l'on désigne soit par le photostyle soit en en tapant le nom au clavier (compte tenu des ambiguïtes possibles, l'unité doit se trouver sur le niveau immédiatement inférieur dans le sous arbre concerné). Pour remonter la commande NIVEAU NB a pour effet de remonter de NB niveaux.

La visualisation de l'arbre d'appel à l'aide de la commande ARBRE peut servir à deux choses. L'utilisateur (s'il prefere travailler directement sur l'arbre) peut voir où il se trouve actuellement, grâce à une flèche indiquant l'unité courante. S'il veut changer d'unité il déplace la flèche en indiquant un autre noeud de l'arbre. L'arbre permet aussi de voir le nombre de niveaux et la répartition d'unités par branches, ce qui peut être utile pour appliquer des fonctions de transformation que nous verrons par la suite. C'est pour cette raison que nous avons implémenté la commande ZOOM PO affichant l'intérieur de l'unité désignée.

III.3.2.2 Orientation

La deuxième famille de fonctions permet de mieux s'orienter sur le schéma.

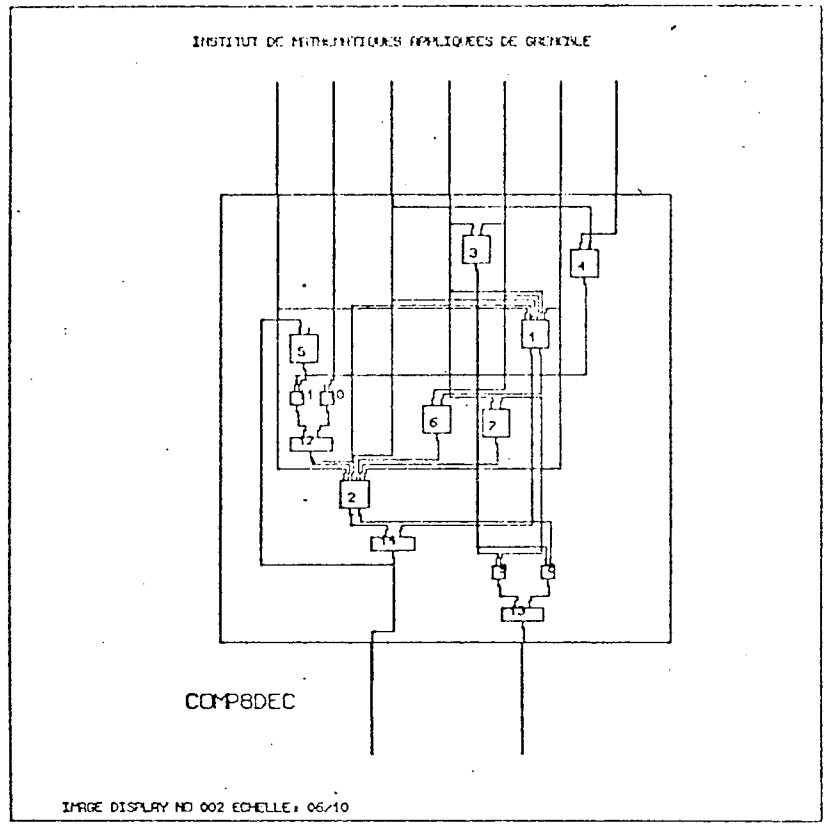
La fonction TEXT donne les noms de toutes les unités sur le schéma.

La fonction NOM PO donne le nom de l'unité designée.

```
****  COMP8DEC

  1 == COMP4(2)
  2 == COMP4(1)
  3 == EXP00000(1)
  4 == EXP00000(0)
  5 == EXP00001(0)
  6 == EXP00002(1)
  7 == EXP00002(0)
  8 == /AND(3)
  9 == AND(2)
 10 == /AND00000(1)
 11 == AND00000(0)
 12 == BUS00000
 13 == BUS00001
 14 == BUS00002


  1 == COMP8DEC
  2 == COMP4
 16 == EXP00000
 17 == EXP00001
 18 == EXP00002
 19 == EXP00003
 20 == EXP00003
 21 == EXP00004
 22 == EXP00005
 23 == EXP00006
 24 == EXP00007
 25 == EXP00007
 26 == AND
 27 == AND
 28 == AND00000
 29 == AND00000
 30 == AND00000
 31 == AND00000
 32 == AND00000
 33 == XPAD0000
 34 == XPAD0000
 35 == XPAD0000
 36 == REG00000
 37 == BUS00000
 38 == BUS00001
 39 == BUS00002
 40 == BUS00003
  3 == COMP4
 41 == EXP00000
 42 == EXP00001
 43 == EXP00002
 44 == EXP00003
 45 == EXP00003
 46 == EXP00004
 47 == EXP00005
 48 == EXP00006
 49 == EXP00007
 50 == EXP00007
 51 == AND
 52 == AND
 53 == AND00000
 54 == AND00000
 55 == AND00000
 56 == AND00000
 57 == AND00000
 58 == XPAD0000
 59 == XPAD0000
 60 == XPAD0000
 61 == REG00000
 62 == BUS00000
 63 == BUS00001
 64 == BUS00002
 65 == BUS00003
  4 == EXP00000
  5 == EXP00000
  6 == EXP00001
  7 == EXP00002
  8 == EXP00002
  9 == AND
 10 == AND
 11 == AND00000
 12 == AND00000
 13 == BUS00000
 14 == BUS00001
 15 == BUS00002
```



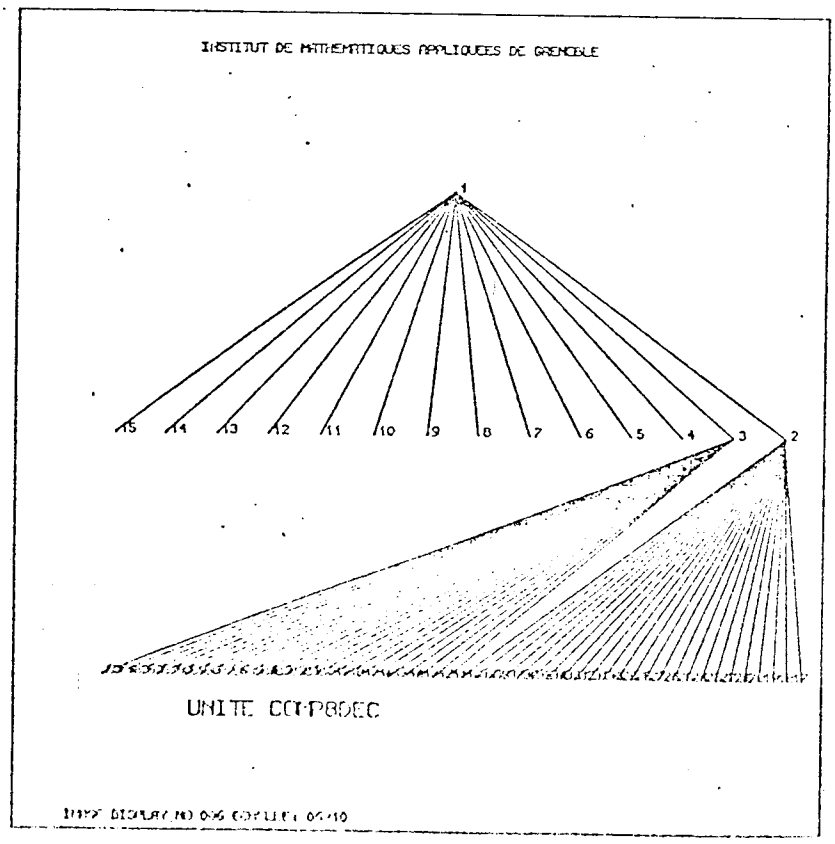


FIG. III-13 COMP8DEC
Schéma et arbre

La fonction inverse EMPLACEMENT NOM montre où se trouve l'objet nommé.

Enfin la fonction DIMENSION /NOM, PO/ donne les dimensions du signal indiqué.

III.3.2.3 Aspect dessin (Fig.III-14)

La troisième famille de fonctions permet l'amélioration de l'aspect graphique du schéma. Nous avons vu que, dans certains cas, notre méthode d'allocation spatiale et de tracé n'est pas suffisamment puissante pour fournir un résultat satisfaisant. C'est pourquoi nous donnons la possibilité de modifier le schéma à l'aide des deux fonctions suivantes:

-- DEPLACER PO PO déplacer l'unité désignée à l'endroit indiqué.

-- CONNEXION PO permet de redessiner la connection tout en gardant le même point de départ et d'arrivée.

III.3.2.4 Restructuration (Fig.III-15)

La quatrième famille de fonctions permet la restructuration hiérarchique (du programme et du schéma) qui peut être utile pour:

-- introduire une nouvelle unité correspondant à une nouvelle fonction que l'on veut mettre en évidence.
-- modifier la répartition des unités.

-- procéder à une optimisation lorsque, en analysant le programme généré, on constate des ressemblances entre différentes unités.

-- pour éviter une disproportion trop grande quant au nombre d'unités dans chaque branche (ce qui pourrait créer des problèmes lors de la visualisation d'une branche trop importante).

C'est pourquoi nous implémentons deux fonctions effectuant la création ou la suppression d'un niveau.

Pour l'opération de création d'un niveau (qui est en fait un regroupement d'unités) il faut désigner les unités qui vont la constituer et donner le nom de l'unité ainsi créée. Le traitement consiste à déterminer les signaux d'entrée et de sortie, à leur donner un nom s'ils n'en avaient pas dans la description d'origine, à trouver le corps de l'unité et à modifier l'arbre en conséquence.

Pour l'opération de suppression de niveau on donne le nom de l'unité à faire disparaître: on remonte au niveau où se trouve l'unité que l'on veut supprimer celles qu'elle contient et on modifie l'arbre en conséquence.

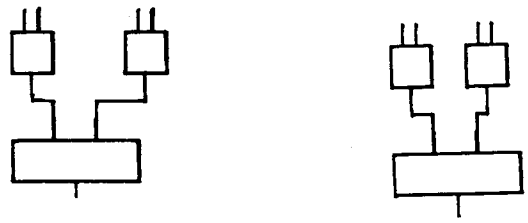
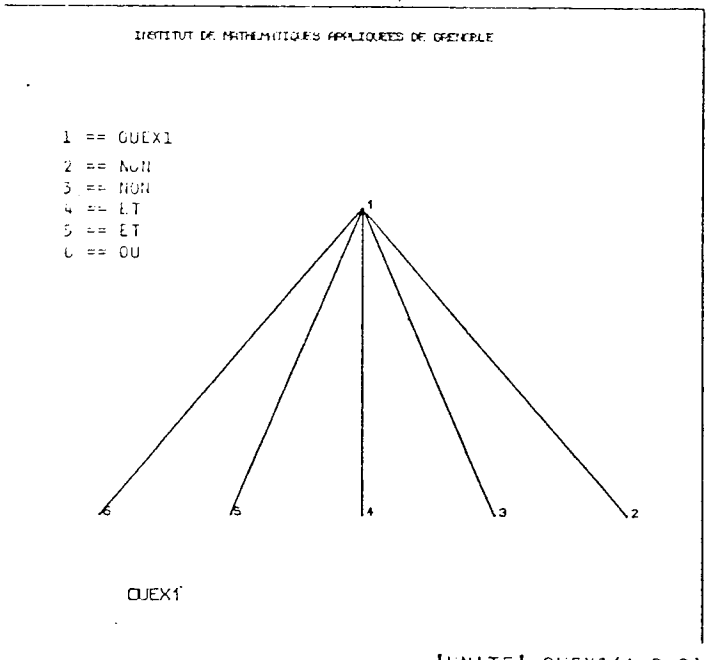
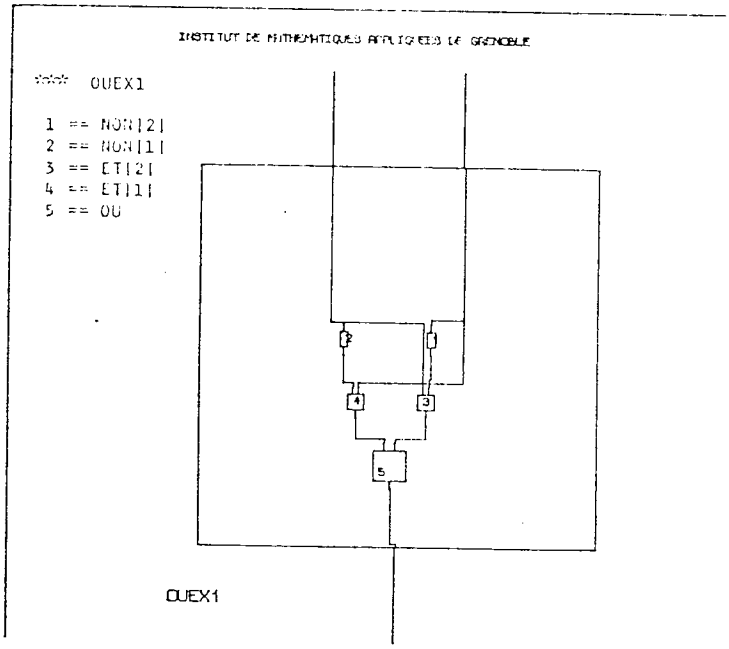
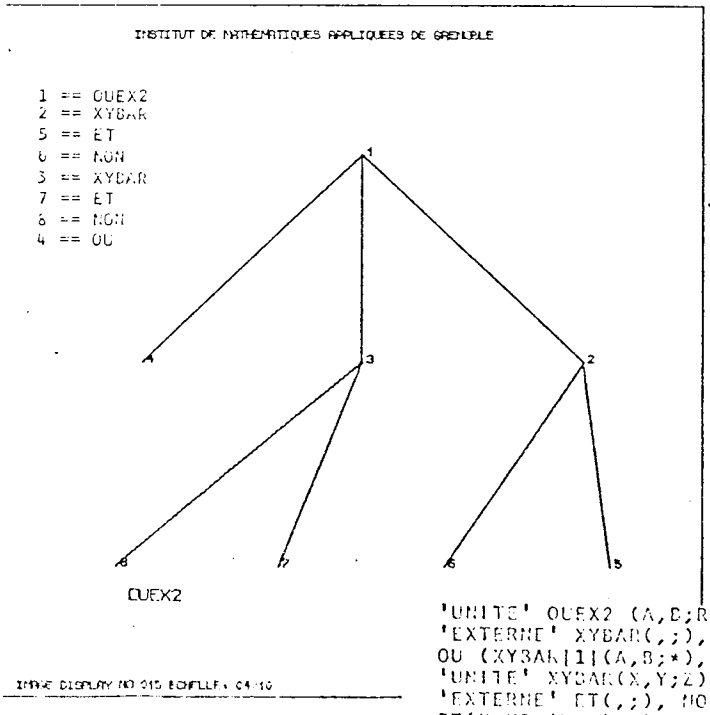


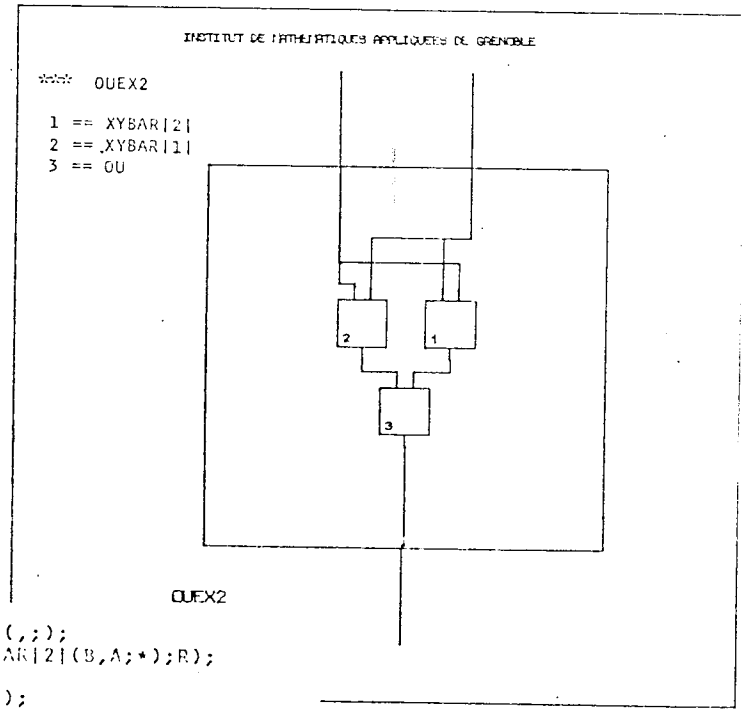
FIG. III-14 Aspect dessin



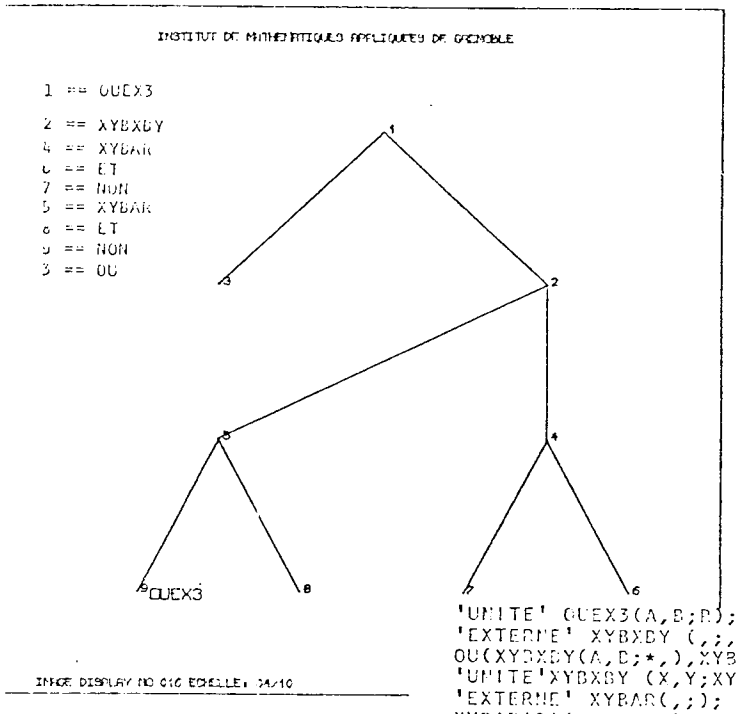


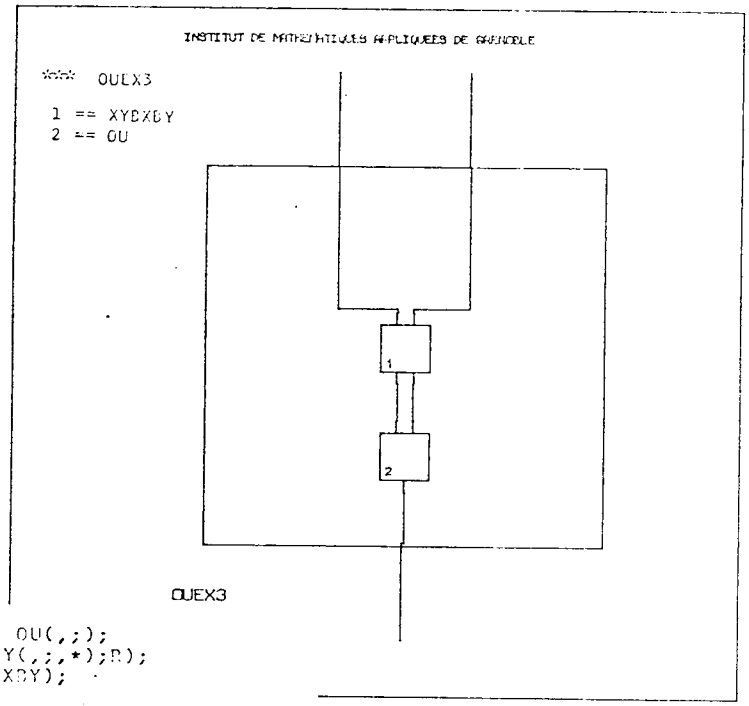

```
'UNITE' OUEX1(A,B;R);
'EXTERNE' NON(;), ET(,;), OU (,;);
OU (ET(NON|1|(A;*),B;*),ET(A,NON|2|(B;*);*);R);
```



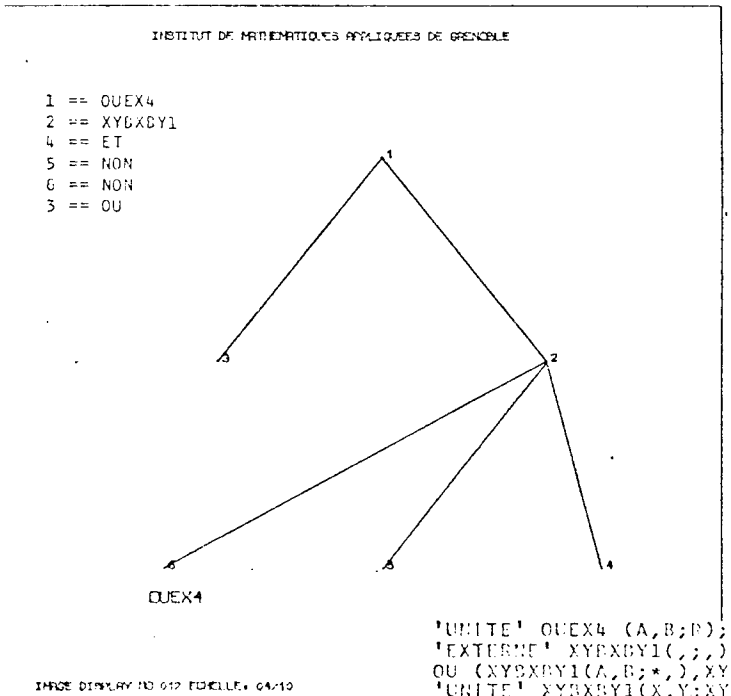


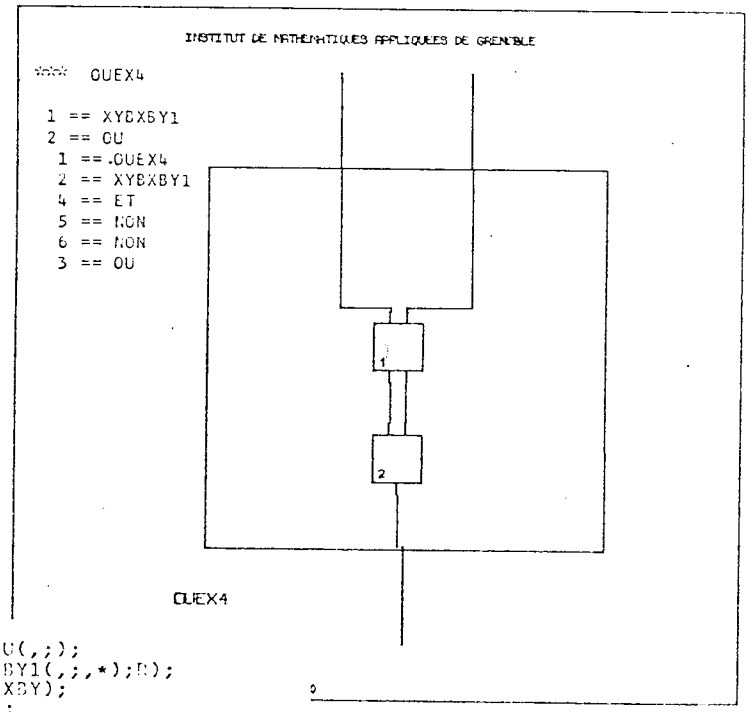

```
'UNITE' OUEX2 (A,B;R);
'EXTERNE' XYBAR(,;), OU(,;);
OU (XYBAR|1|(A,B;*),XYBAR|2|(B,A;*);R);
'UNITE' XYBAR(X,Y;Z);
'EXTERNE' ET(,;), NON(;);
ET(X,NON(Y;*);Z);
```






```
'UNITE' OUEX3(A,B;R);
'EXTERNE' XYBXBY (,;,), OU(,;);
OU(XYBXBY(A,B;*,),XYBXBY(,;,*);R);
'UNITE'XYBXBY (X,Y;XYB,XBY);
'EXTERNE' XYBAR(,;);
XYBAR|1|(X,Y;XYB); XYBAR|2|(Y,X;XBY);
```






```
'UNITE' OUEX4 (A,B;R);
'EXTERNE' XYBXBY1(,;,),OU(,;);
OU (XYBXBY1(A,B;*,),XYBXBY1(,;,*);R);
'UNITE' XYBXBY1(X,Y;XYB,XBY);
'EXTERNE' ET(,;), NON(;);
ET(X,NON|1|(Y;*);XYB); ET(Y,NON|2|(X;*);XBY);
```

FIG. III-15 Différentes structurations d'un OU Exclusif

Remarques: Le problème que l'on peut rencontrer est celui de la conformité des signaux:

-- dans la description d'origine on aurait pu utiliser le même nom de signal dans les deux niveaux sans pour autant désigner la même chose. Si on supprime le niveau on aurait donc deux signaux logiquement différents portant le même nom.

-- lors de la création d'une unité on doit souvent générer des signaux; lors de la suppression il faut faire disparaître tous les signaux générés qui ne sont pas nécessaires, sinon, au bout de quelques opérations, la description ne ressemblerait plus à ce que l'on avait à l'origine.

Nous effectuons ces opérations directement sur le langage ce qui nous permet de les utiliser sans terminal graphique et, par là, pour des applications dans lesquelles on ne travaille pas avec les schémas.

Nous donnons à l'utilisateur la possibilité de grouper des commandes et de restructurer ainsi plus rapidement le programme.

III.3.2.5 Modifications fonctionnelles

La cinquième famille permet les modifications fonctionnelles de la description. Il faut pouvoir agir sur les unités et sur les connections: éliminer ou bien au contraire faire intervenir de nouvelles unités et indiquer leurs interconnections.

Les fonctions de modification sont donc les suivantes:

-- pour les modifications de connection:

-- EFFACER CONNEXION PO.

-- DEFINIR CONNEXION PO PO en indiquant le point de départ et d'arrivée de la connection.

Remarque: Pour pouvoir définir une nouvelle connection il faut que les deux parties à relier soient compatibles du point de vue dimensionnel. L'utilisateur peut s'en assurer avec la commande DIMENSION. Si on avait à sa disposition différents types de connections (chemin de données, chemin de contrôle, signal de type niveau ou impulsion) on pourrait faire des vérifications de conformité plus poussées.

-- pour les unités:

-- ENLEVER PO en désignant l'unité à faire disparaître.

-- PLACER NOM en indiquant le nom de l'unité à entrer.

- -CHOISIR en montrant dans le catalogue l'unité à entrer

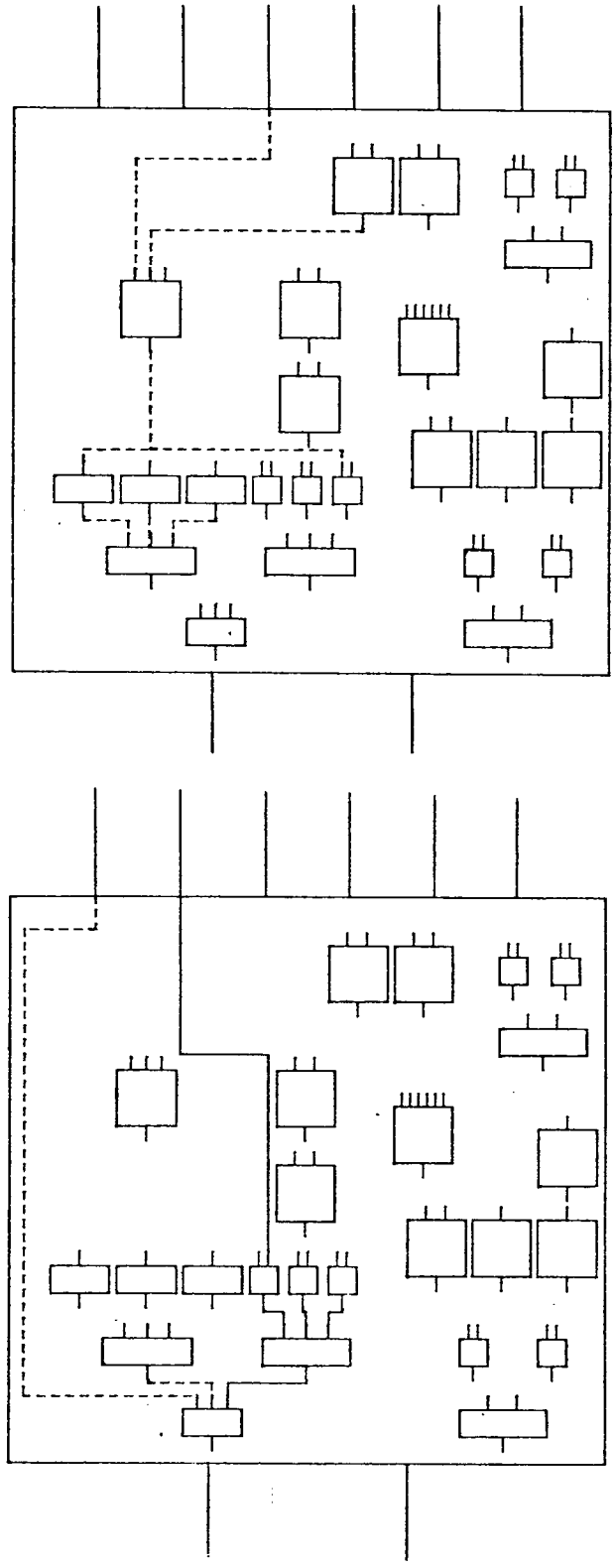


FIG. III-16 Constructions fonctionnelles

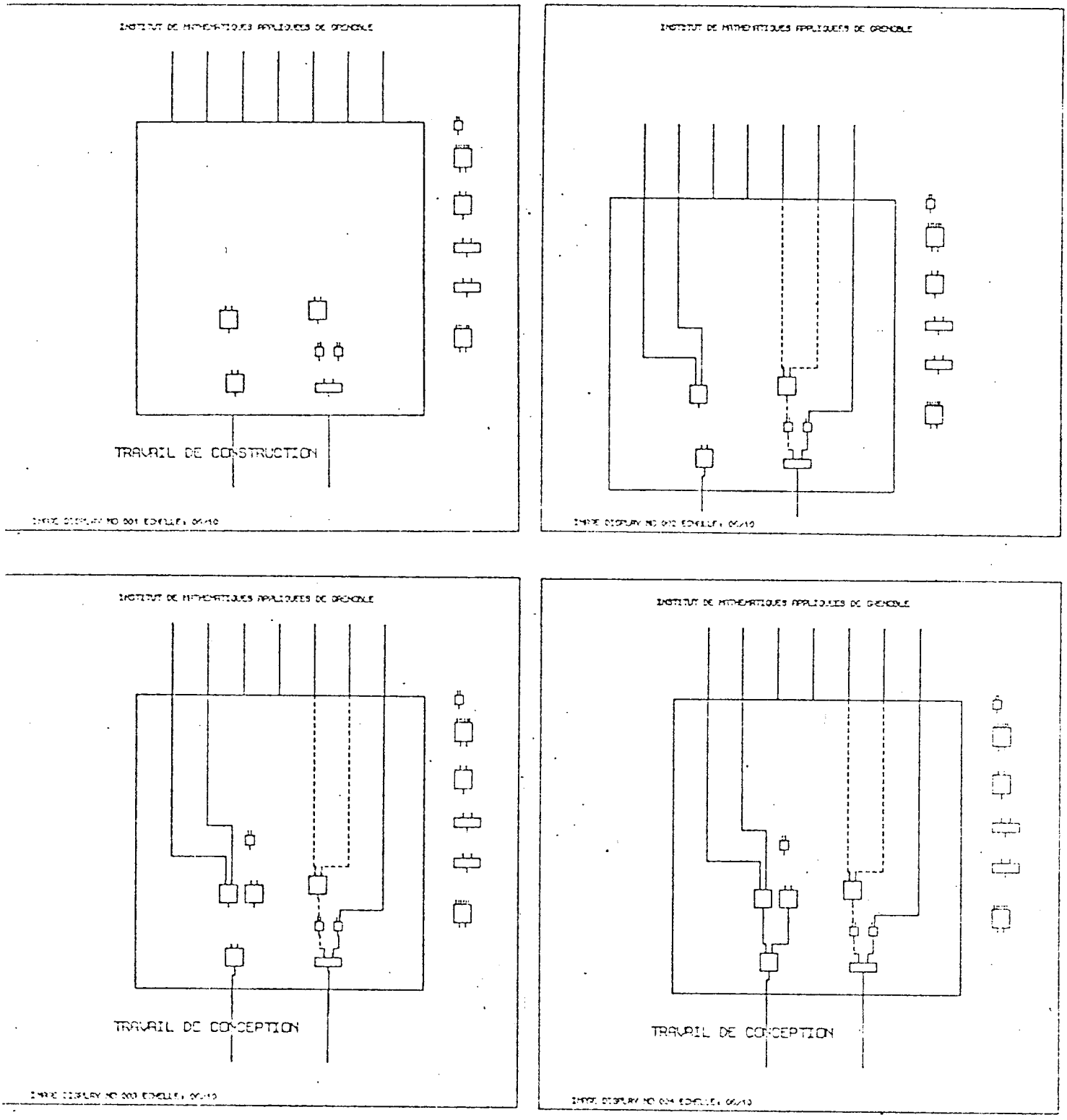


FIG. III-17 Travail de conception

Ici il faut prendre des options quant aux modifications que l'on peut effectuer sur la famille d'unités primitives. Les restrictions peuvent être plus ou moins sévères:

-- imposer que l'ensemble d'unités primitives soit fermé.

-- donner la possibilité à l'utilisateur d'introduire de nouvelles unités primitives ou même de se définir son ensemble personnel de primitives.

Cette deuxième approche, intéressante surtout pour l'enseignement, nécessite l'utilisation de l'environnement de création.

Remarque: On doit restreindre les possibilités de modifications fonctionnelles des unités utilisées à plusieurs endroits.

III.3.3 Environnement d'analyse

Pour apprécier la description il faut pouvoir la juger aussi bien d'un point de vue statique (structurel) que dynamique (fonctionnel).

La vérification fonctionnelle se fait par simulation (synchrone ou asynchrone). Le passage dans l'environnement d'analyse fait appele à une transformation qui produit à partir de la description en CASSANDRE un modèle exécutable. L'utilisateur commande

la simulation directement à partir du terminal graphique. Les commandes proprement de simulation sont transmises directement au programme de simulation; quant aux commandes de positionnement et d'interrogation du simulateur on les interprète aussi pour la visualisation: quand un positionnement à l'intérieur d'une unité est demandé, on affiche le schéma sur le terminal; quand on interroge, la réponse apparaît sur le schéma.

Il serait souhaitable:

-- pour un jugement structurel de donner un aperçu sur le matériel nécessaire (si on a donné le matériel nécessaire à chaque unité primitive).

-- de donner des statistiques quant à l'utilisation de différentes unités primitives ou non primitives.

-- de donner l'évolution dynamique du réseau et de l'arbre, donc une suite de chemins et d'unités actives à un moment donné.

Remarque: Deux derniers points peuvent servir à optimiser l'utilisation du matériel, ce qui peut être fait en revenant par exemple dans l'environnement de modélisation.

III.3.4 Environnement de sortie

Un autre environnement appelé SORTIE a pour but de regrouper toutes les opérations de sorties. C'est dans cet environnement que l'utilisateur peut demander:

-- la sortie sur table traçante du schéma courant ou de l'arbre d'appel.

-- la sauvegarde, sous forme interne, de l'etat courant de la modélisation, ceci pour permettre des retours en arrière.

-- générer la nouvelle description en langage, pour les unités qu'il vient de modifier.

Après avoir indiqué ces options il peut quitter le système.

III.4 Organisation du système

Nous terminons ce chapitre par une présentation succinte de la mise en oeuvre de la partie du système CASSANDRE que nous avons réalisée.

L'aspect fonctionnement a été largement commenté dans les pages précédantes, nous donnons donc uniquement le schéma de fonctionnement (Fig. III-18). Pour les commandes nous avons choisi d'utiliser au maximum les touches de fonctions et le photostyle et très rarement le clavier du terminal graphique.

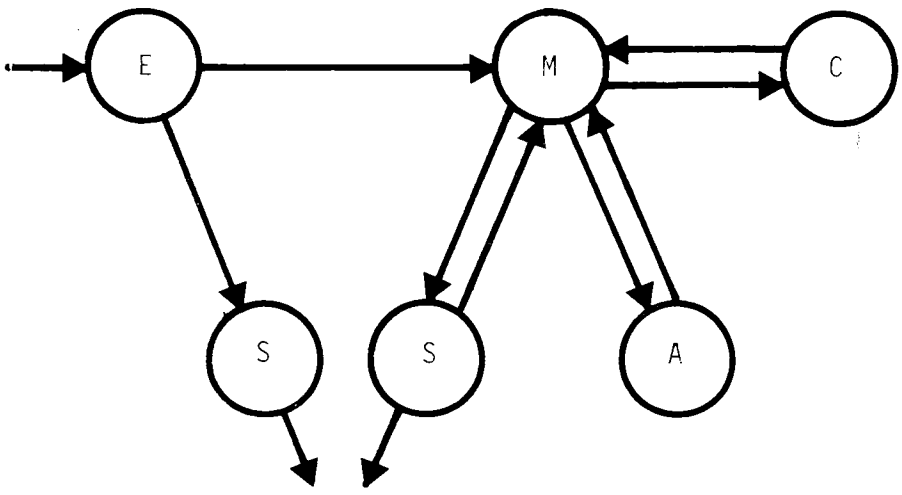


FIG. III-18 Schéma du fonctionnement

L'aspect gestion des données est représenté sur la fig.III-19. La gestion par fichiers nous semble valable car les données et les traitements ne sont pas très variés. Nous avons ajouté trois types de fichier:

-- fichier CATALOGUE contenant les noms des unités cataloguées

-- fichier ARBRE traduisant la hiérarchie des unités dans un travail donc un seul fichier par projet

-- fichier TOPSTRUC contenant la description structurelle et topologique d'une unité donc un fichier par unité.

La description externe (en langage) est vérifiée et traduite dans une description interne équivalente où on sépare la description des objets en présence de leurs relations. Nous analysons ces descriptions afin d'obtenir les données nécessaires pour les traitements de construction de schémas, puis le schéma constitué est stocké dans un fichier du type TOPSTRUC, sur lequel s'effectuent les manipulations interactives.

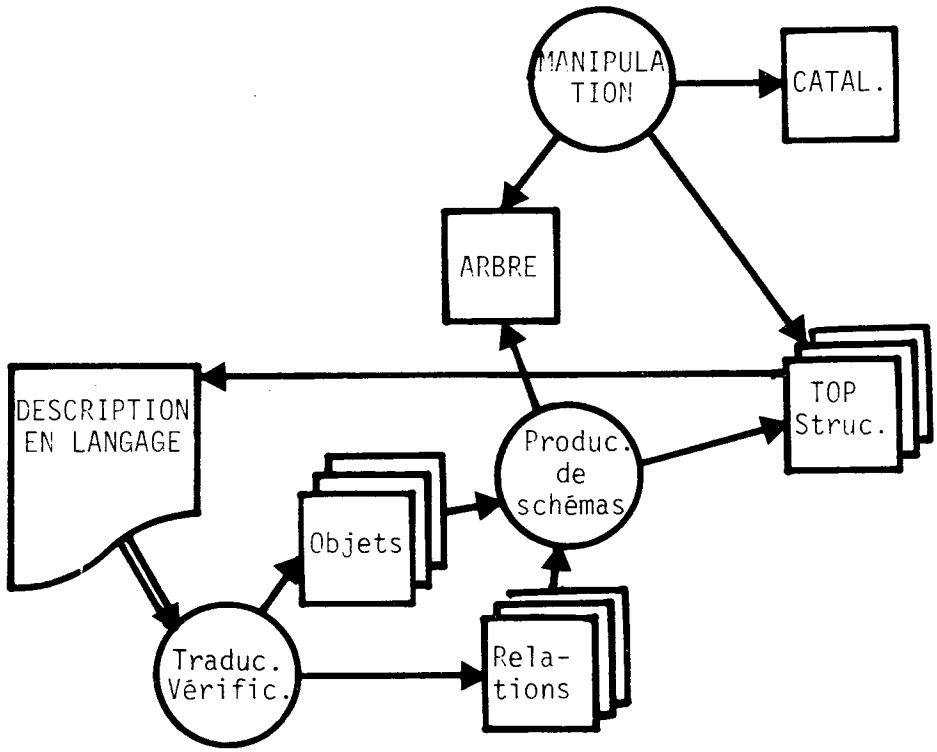


FIG. III - 19 Organisation du système pour les manipulations de schémas logiques

Bibliographie

|1| AKERS SB
ROUTING in : Design Automation of Digital Systems / pp.283-333
Prentice Hall / 1972

|2| BRESSY Y., DAVID B., FANTINO Y., MERMET J.
A hardware Compilation for Interactive Realisation of Logical Systems described in CASSANDRE
Workshop on Computer Hardware Description Languages and Their Applications / New York / sept. 3-5 / 1975

|3| BREUER MA
Design Automation of Digital Systems
Prentice Hall / 1972

|4| DAVID B., DELRIEU
Etude bibliographique de techniques de placement et de trace
Rapport interne / ENSIMAG / 1974

[5] DAVID B., FANTINO T., MENARD G
Outils pour le decoupage en module, l'implantation et la documentation
automatique de machines digitales decrits en CASSANDRE
Rapport interne / ENSIMAG / 1973

[6] DE POLIGNAC K
Utilisation du langage CASSANDRE pour la conception de machines microprogrammees
These de 3e cycle / USMG / 1973

[7] EASTMAN C
Representation for Space Planning
CACM / Vol.13 / no.4 / pp.242-250 / 1970

[8] FARRENT H
Edition automatique de schemas logiques
These Docteur-Ingenieur / Toulouse / 1971

[9] GRIFFITHS M., PELTIER P.
Grammar transformation as an aid to compiler production
Etude / Centre scientifique IBM France / 1968

[10] HANAN M., KURTZBERG JM
PLACEMENT TECHNIQUES in : Design Automation of Digital Systems
/ pp.213-282 /
Prentice Hall / 1972

[11] LIDDELL P
Decoupage syntaxique de systemes logiques decrits en CASSANDRE
These de 3e cycle / USMG / 1970

[12] LUSTMAN F
Simulation d'une machine digitale a partir d'une description en langage CASSANDRE
Revue bleue de l'AFIRO / no.B-2 / 1969

[13] MERMET J
Definition du langage CASSANDRE
These de Docteur Ingenieur / USMG / 1970

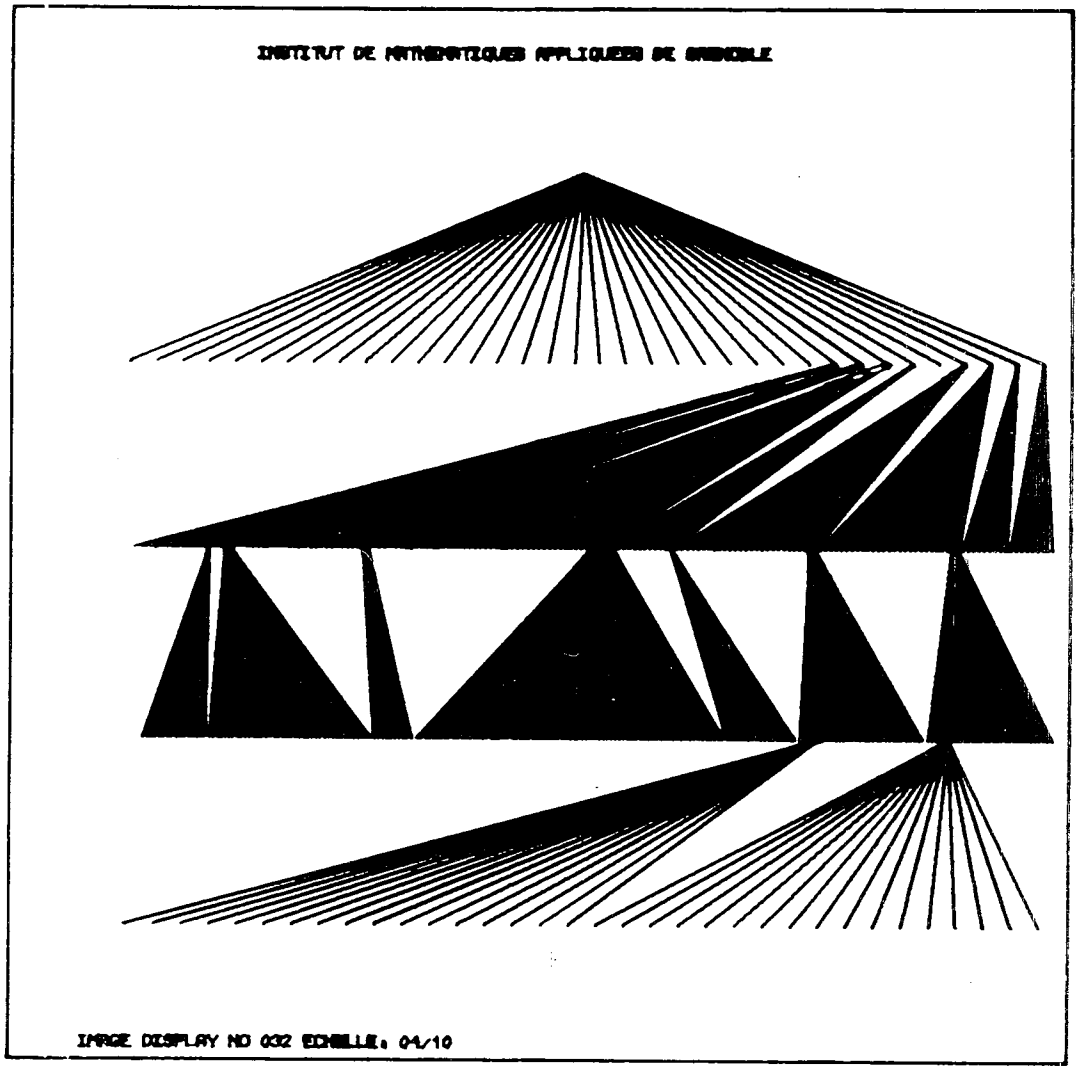
[14] MERMET J
Etude methodologique de la conception assistee par ordinateur du systemes logiques: CASSANDRE
These d'etat / USMG / 1973

[15] SAILLARD JC., SARRET M
Dessin automatique des masques : Le programme DESMAG
Onde electrique / Vol.49 / no.502 / pp.92-97 / 1969

[16] SARRET MJ
Problemes d'implementation : le programme DESMAG
These 3e cycle / USMG / 1969

[17] STEVENS JL jr
Fast Heuristic Techniques for Placing and Wiring Printed Circuit Boards
Phd Thesis / Center for Advanced Computation / University of Illinois / 1972

[18] VANCLEEMPUT W., LINDERS JC
An Interaction system for Computer-Aided Design of Printed Circuit Boards
Proc : ACM / 25th anniversary conf. 1972 / pp.390-397 / 1972



CHAPITRE IV

APPLICATION: ARCHITECTURE

IV.0 Avertissement

Nous travaillons depuis presque trois ans en collaboration avec les architectes de l'Unité Pédagogique d'Architecture de Grenoble sur les possibilités d'utilisation de la C.A.O. en architecture.

Après un bref aperçu méthodologique et bibliographique du contexte de l'application, nous présentons un modèle du processus architectural explicitant la démarche de l'architecte et permettant donc une conception assistée par ordinateur.

Puis nous présentons le système SIGMA-ARCHI: un système interactif graphique pour méthodes d'aide à la conception architecturale, basé sur ce modèle.

Après une description fonctionnelle nous montrerons qu'il s'agit d'une utilisation particulière du système SIGMA-C.A.O. dans le domaine de la conception architecturale.

C'est la description détaillée de deux phases concernant l'allocation spatiale et la construction interactive de plans de construction qui termine le chapître.

IV.1 Contexte de l'application

Bien qu'ayant à resoudre des problèmes inconnus jusqu'alors, les architectes continuent, d'une maniere qui leur semble normale, à utiliser les méthodes traditionnelles dont certaines paraissent désuètes; devant l'accroissement des demandes (23), un certain nombre de conséquences en découlent: augmentation du nombre d'architectes ou prolifération de constructions sans leur participation, impossibilité d'aborder les problèmes de maniere suffisamment globale (c'est-à-dire en tenant compte des considérations démographiques, sociologiques et autres), etc....

Dans cet esprit, on assiste à un intérêt croissant pour la C.A.O. en architecture.

L'introduction de l'ordinateur dans ce domaine doit être faite avec le plus de précautions possibles pour ne pas effrayer les architectes et ne pas se heurter à un refus categorique. Il faut bien montrer que le but est de faciliter le travail de l'architecte, en le débarassant des tâches quotidiennes qui peuvent être accomplies par ordinateur.

Puis on essaie d'utiliser l'ordinateur à des tâches plus nobles, en lui proposant un travail plus complexe; il devient ainsi un véritable collaborateur. On essaie alors d'améliorer les échanges homme-machine en recherchant des formes et des moyens appropriés, commodes et simples.

Il faut necessairement introduire la communication graphique si proche des supports de travail habituels de l'architecte.

IV.1.1 Aspect methodologique

Afin d'arriver à déterminer ce que peut apporter l'informaticien dans la conception architecturale, nous allons passer en revue un certain nombre de methodes qui y sont employées.

Parmi les instrumentations utilisées en architecture, d'après Zeitoun (50), nous en avons retenu les trois suivantes.

-- analyse synthétique des contraintes. Indispensable pour dominer une grande quantité de données. On l'appelle analyse de donnees.

-- outils pour la présentation et la manipulation de données. Il s'agit de tous les types de visualisation geometrique, de représentation et de dessin servant de support au processus de conception architecturale.

-- allocation spatiale: distribution des élements aussi bien dans le temps que dans l'espace.

IV. 1. 1 Pour une généralisation des systèmes C. A. O.

Le comportement de l'informaticien vis-à-vis de ces instrumentations et vis-à-vis de l'architecte doit être très prudent et réfléchi. Il doit surtout avoir toujours à l'esprit qu'il ne peut qu'aider et non remplacer: il propose des outils et des méthodes, et doit toujours laisser à l'architecte la possibilité de les juger et de justifier leur place dans le processus de conception: seul l'architecte a compétence pour l'interprétation des résultats de la résolution.

IV. 1. 2 Aspect bibliographique

Il nous paraît très difficile de faire rapidement le point sur un domaine aussi vaste que la C. A. O. en architecture. Même en ne s'intéressant qu'aux différents mécanismes de résolution de problèmes nous constatons une diversité considérable: décomposition hiérarchique, graphe dual, projection, simulation, jeu, chaîne de Markov, etc...

L'aspect bibliographique a été étudié dans le cadre de nos recherches par Rivero (43); il fait le point sur la situation de 1973. Actuellement nous ne pouvons que confirmer ses propos.

Enumérons les types de recherche:

-- analyse de données: constitue un des domaines de travail le plus fructueux. Originellement basées sur les travaux d'Alexander, les méthodes d'analyse de données ont beaucoup progressé, en se diversifiant. On trouve dans (13) un essai de classification de ces méthodes et dans (11) leur utilisation dans le processus de conception architecturale. Les publications les plus significatives sont (6), (21), (30).

-- allocation spatiale: problème d'arrangement d'un ensemble d'éléments. Les méthodes existantes sont nombreuses; inspirées du programme CRAFT (2), elles s'appliquent toutes dans un contexte particulier avec des données prises en compte très limitées. (9) est un excellent article qui fait le tour des méthodes et de leurs possibilités. Les méthodes types se trouvent dans (4), (18), (40), (41).

-- approche particulière: on la trouve dans les travaux de l'équipe de Negroponte (33), (34), (35). Il s'agit de la construction de périphériques particuliers servant de moyens de saisie et de communication (reconnaissance de schéma, ...). On peut ainsi résumer cette démarche: adapter l'informatique à la pratique architecturale et non pas aborder des problèmes architecturaux à l'aide d'outils informatiques habituels.

-- systèmes intégrés: leur but est de suivre le processus de conception et de construction en donnant à chaque fois les outils nécessaires (22), (23). Nous avons surtout remarqué le projet ARK2 (27). Ils sont souvent orientés vers une conception modulaire.

-- outils très opérationnels: on les trouve surtout dans la phase d'aide à la construction; ils se rencontrent essentiellement en Grande Bretagne (10), (38), (39).

Nous donnons dans IV.7 une liste des principaux centres travaillant dans le domaine de la C.A.O. en architecture et la liste des principales manifestations traitant ce sujet.

IV.2 Modèle de conception

L'obstacle principal au développement des outils C.A.O. pour l'architecte est le manque de connaissances sur le but de son activité et sur son activité elle-même. En effet, les travaux sur la méthodologie architecturale sont très récents et font encore l'objet de grandes controverses. Ces travaux ont pris de l'importance à partir de la publication en 1964 de "Notes on the Synthesis of the Form", de l'architecte et mathématicien Anglais Christopher Alexander.

Les travaux d'Alexander, joints à ceux de Friedmann, Negroponte, Bernholtz, Eastman, Rittel, pour n'en citer que quelques uns, ont permis d'éclaircir quelque peu ce qui jusqu'alors était considéré comme un processus de créativité, de nature fondamentalement intuitive et non rationalisable.

Ces nouvelles lumières ont montré des phases, voire des tâches bien définies qui pourraient dans certains cas, être automatisées et soumises à l'ordinateur.

Les taxonomistes architecturaux se sont tout de suite attaché à classer et nommer ces phases et malgré un manque de consensus, on peut arriver au modèle général suivant:

-- formulation

-- organisation spatiale

-- modularisation

-- allocation spatiale

-- instrumentation

-- traitements documentaires

La formulation est l'énoncé du futur contenu socio-culturel du projet, en ce qui concerne les activités, l'ambiance, les us et coutumes. C'est une phase de spécification du projet qui produit comme résultat un "Programme Architectural".

L'organisation spatiale permet de caractériser les "espaces" qui contiendront les activités en fonction des propriétés (surface, éclairage, isolations acoustique et thermique, etc...) requises pour le déroulement normal de l'activité. Par activité on entend tout processus qui est ou qui peut être réalisé dans un lieu précis, indépendamment de tout autre processus (manger, se reposer, lire, etc...). Le résultat de la phase d'organisation spatiale est donc une liste d'espaces qualifiés par leurs propriétés et par ses relations avec les autres espaces.

La modularisation est une phase optionnelle, nécessaire lorsque pour des raisons économiques on veut utiliser des composantes industrielles standardisées ou/et réduire le nombre de types de composantes à utiliser (dans un projet spécifique).

L'allocation spatiale a pour but la génération d'une configuration, d'un arrangement des espaces, satisfaisant aux relations définies lors de la phase d'organisation spatiale.

L'instrumentation consiste à déterminer l'habillage matériel du ou des enveloppes résulant de la phase précédente.

Le traitement documentaire est la génération de tous les documents nécessaires à la production de l'objet conçu: devis, plans, etc...

La linéarité du schéma proposé n'implique pas la linéarité du processus de conception basé dessus; nous avons seulement indiqué la direction générale de l'évolution de la conception. Cette dernière résulte en des allées et venues entre les différentes phases: il est évidemment possible de remettre en question n'importe quelle phase déjà terminée, aussi bien que de réitérer en revenant à n'importe quelle phase.

IV.3 Système SIGMA-Archi

IV.3.1 Traitement des informations

Pour pouvoir réaliser un système à partir du schéma précédemment présenté, il faut constituer un support pour tous les types d'information qui peuvent apparaître au cours du processus de conception.

Trois types d'information sont facilement reconnaissables:

-- informations logiques ou relationnelles: elles apparaissent essentiellement au début du processus, comme résultat de la formalisation et de l'analyse.

-- informations topologiques ou géométriques: elles sont générées, en partie pendant l'analyse de données, par la transformation d'informations relationnelles, et en partie pendant l'allocation spatiale.

-- informations techniques: elles interviennent surtout en fin de processus lors de la spécification définitive.

Tous les types d'information peuvent être spécifiés pendant n'importe quelle phase: si certaines spécifications sont imposées, ceci est indispensable.

Cette constation sur la disponibilité des informations peut être étudiée sur toute la vie d'un objet architectural. Les liens entre les différentes phases de sa vie peuvent, et c'est très souhaitable, être assez étroits. Cela permettrait, par exemple, d'entreprendre les modifications de l'objet en respectant la philosophie de sa conception. On peut traduire ceci à l'aide du schéma suivant (Fig. IV-1):

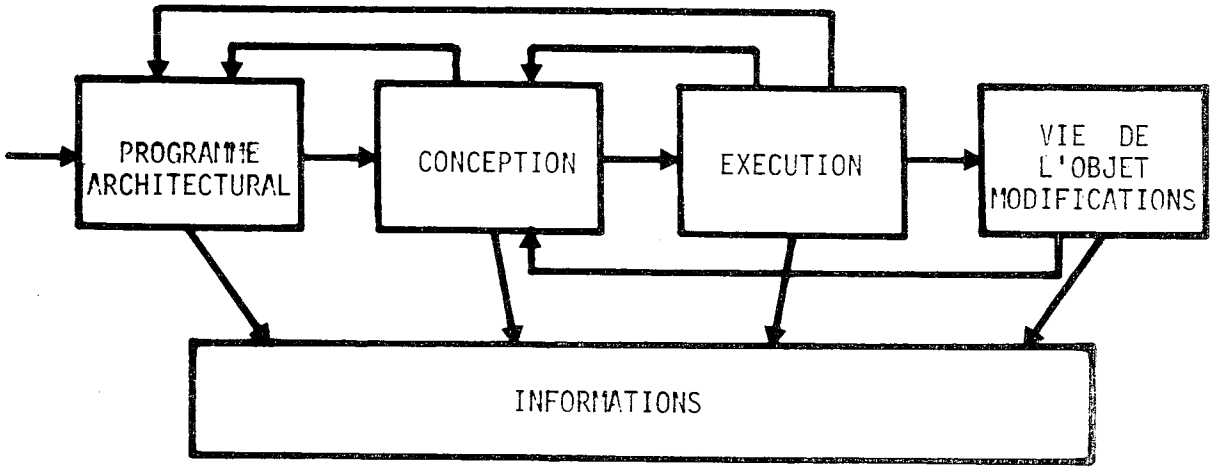


FIG. IV-1 La vie d'un objet architectural

Mais revenons au problème de la conception architecturale pour préciser la signification des données que l'on prend en compte dans le processus.

Les données logiques concernent les unités de conception et leurs relations; les données géométriques concernent d'une part les formes brutes, produites par les méthodes d'allocation spatiale, et d'autre part les formes finies dans lesquelles on fait intervenir des données techniques (solutions techniques, équipements).

Le système SIGMA-Archi, que nous présentons ici, s'est donné comme but de constituer un support actif et permanent pour le processus architectural; comme son nom l'indique, Système Interactif Graphique pour les Méthodes d'Aide à la conception ARCHItecturale, il doit être interactif, graphique, et fournir des aides. Ces trois objectifs nous paraissent très importants car: seul un dialogue étroit peut permettre une véritable collaboration; le graphique est un moyen d'expression proche de l'architecte et les méthodes d'aides peuvent, si l'architecte le désire, lui permettre d'augmenter ses possibilités.

Le système a pour but de générer les données architecturales que nous avons schématisées précédemment, et de permettre leur progression pour aboutir à un objet architectural satisfaisant et faisable.

Compte tenu de ce qui précède, nous pouvons schématiser le flot des informations dans le système de la façon suivante (voir Fig.IV-2):

L'utilisateur peut, à n'importe quel moment, entrer les informations et les sortir sous une forme souhaitée.

Il peut aussi effectuer d'une façon entièrement interactive des manipulations, pour faire progresser le processus de conception. Dans la phase logique il s'agit surtout d'étudier et d'organiser les données disponibles, dans la phase concernant l'obtention de formes brutes il les construit en tenant compte des données logiques et, éventuellement, des formes brutes déjà obtenues. Dans la phase concernant l'obtention des formes finies, il habille les formes brutes

en tenant compte des informations logiques (fonctionnelles) et en utilisant les catalogues appropriés.

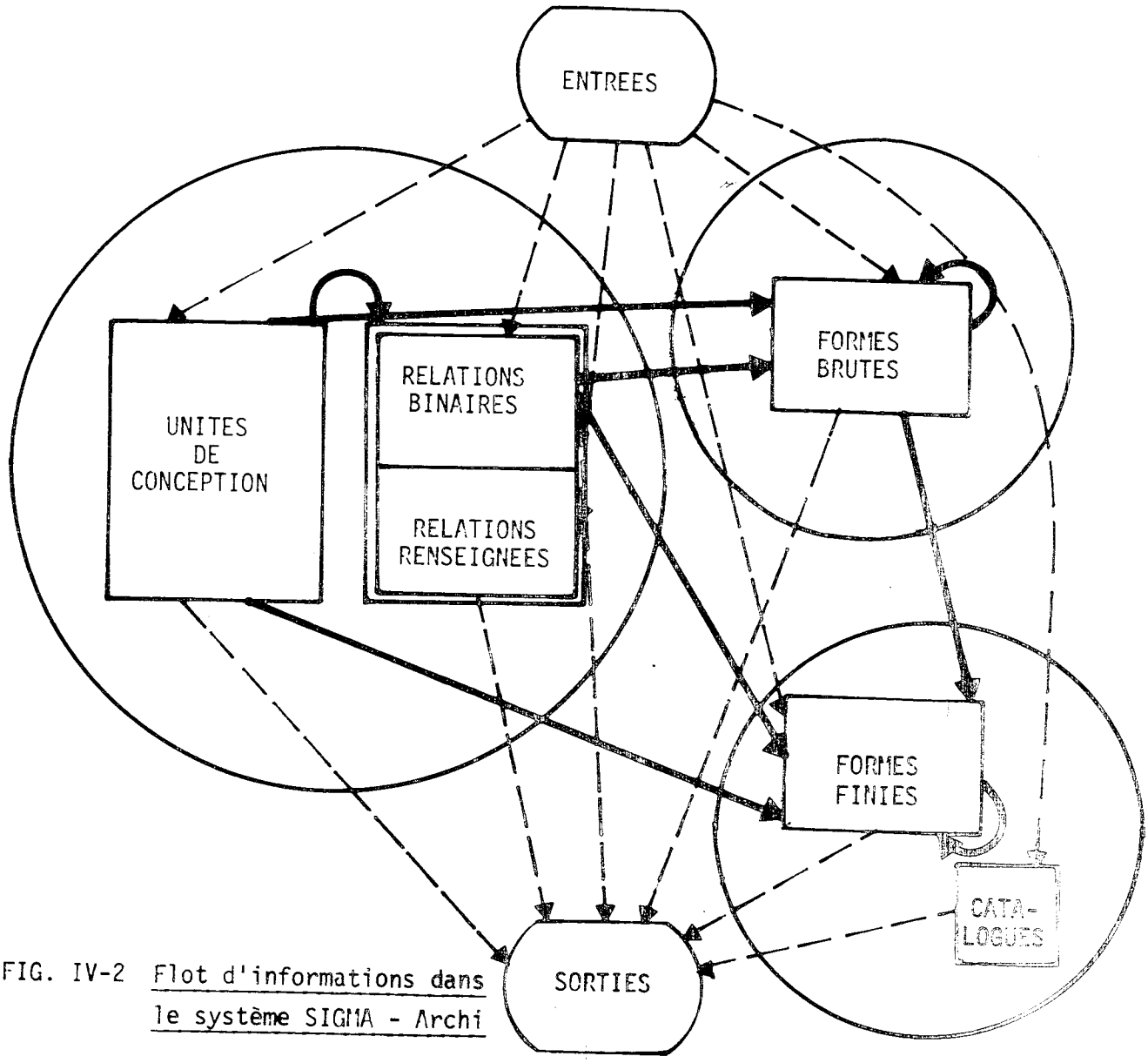


FIG. IV-2 Flot d'informations dans le système SIGMA - Archi

S'il le désire il peut utiliser ou se construire des traitements effectuant les démarches précédentes d'une façon plus automatisée; le degré d'automatisation et le choix des traitements restent évidemment à sa portée, ainsi que la possibilité de reprendre ou remettre en cause des solutions proposées, et les juger et les modifier de façon interactive.

IV.3.2 Exemple: centre socio-culturel

Etudions un exemple pour montrer une possibilité de travail avec le système.

L'exemple choisi est purement pédagogique et la solution proposée ne prétend pas avoir une signification architecturale approfondie. Il s'agit seulement de montrer une méthode de travail possible.

Il s'agit de concevoir un centre socio-culturel dans lequel on veut pouvoir exercer les activités suivantes (Fig.IV-3):

01	COIFFURE COUTURE	14	MUSIQUE
02	PYROGRAVURE	15	CONF.
03	MODELAGE POTERIE	16	JEUX SOCIETE
04	MAQUETTISME	17	JEUX COLLECTIFS
05	SCULPTURE	18	GYMNASTIQUE DANSE
06	PHOTO	19	HALL
07	CUISINE	20	NURSERIE
08	NATURE	21	ENCADREMENT
09	MECANIQUE MENISERIE	22	INFIRMERIE
10	CHIMIE	23	RESTAURANT
11	BIBLIOTHEQUE	24	BUREAU
12	CINEMA	25	PSYCHO
13	THEATRE	26	EXTERIEUR

Le travail de formulation s'effectue d'une façon analytique. On prend le programme architectural et on étudie les activités pour préciser leurs propriétés; puis on dégage les relations entre ces activités.

Puis on passe en revue les activités et on précise leur comportement vis-à-vis des propriétés que l'on considère comme importantes: surface, hauteur, forme, bruit, propreté, qualité du sol, accès extérieurs, points d'eau, etc...

Puis on étudie les relations entre les activités. Ces relations sont de deux sortes: d'une part les relations fonctionnelles fournies par l'utilisateur (accès, proximité, cheminement, etc...) (Fig.IV-4) et d'autre part les relations déduites des propriétés des activités (compatibilité de bruit, compatibilité de hauteur, etc...).

Cette partie qui concerne l'explicitation du programme architectural et qui s'effectue d'une façon analytique, est assistée par un programme interactif de saisie d'information. Pour notre exemple, le résultat se trouve sur la figure IV-5.

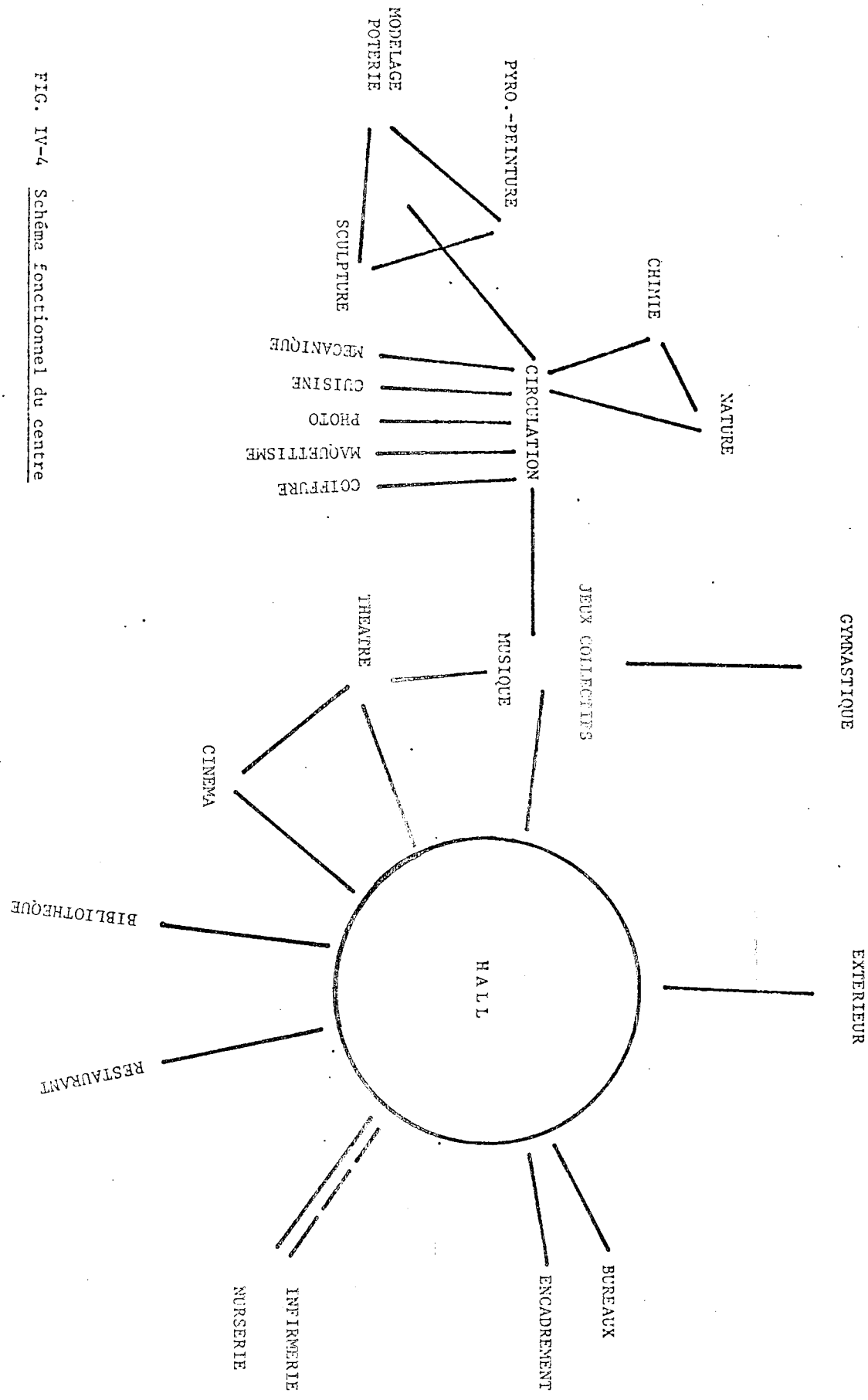


FIG. IV-4 Schéma fonctionnel du centre

	SURFACE	HAUTEUR	BRUIT	PROPRETE	LUMIERE NATURELLE	ACCES EXTERIEUR	EAU
1 . Coiffure - Couture	18	3.50	3	2	2	2	1
2 . Pyrogravure - Peinture	21	3.50	3	1	3	2	1
3 . Modelage - Poterie	18	3.50	3	1	3	0	1
4 . Maquettisme	21	3.50	3	2	2	2	0
5 . Sculpture	18	3.50	4	2.	3	0	0
6 . Photo	10	2.40	3	1	0	2	1
7 . Cuisine	28	3.50	3	1	2	2	1
8 . Nature	18	3.50	3	3	2	0	1
9 . Mécanique - Menuiserie	28	3.50	4	2	2	0	0
10 . Chimie	18	3.50	3	3	2	2	1
11 . Bibliothèque	77	3.50	2	4	2	1	0
12 . Cinéma	83	3.50	1	3	0	1	0
13 . Théâtre	56	3.50	1	3	0	0/1	0
14 . Musique	28	3.50	1	3	2	2	0
15 . Conf. Education Sexuelle	28	3.50	2	3	2	2	0
16 . Jeux Société...	24	2.40	3	3	2	2	0
17 . Jeux Collectifs	110	6.00	4	3	2	0	0
18 . Gymnastique (Danse, Ed.C)	70	3.50	2	3	2	4	0
19 . Hall	104	3.50	3	3	2	0	0
20 . Nurserie	70	3.50	2	4	2	1	1
21 . Encadrement	21	2.40	2	3	2	2	0
22 . Infirmerie	10	2.40	2	4	2	0/1	1
23 . Restaurant	104	3.50	3	3	2	0/1	1
24 . Bureaux	18	2.40	2	3	2	1	0
25 . Psycho.	52	2.40	2	4	2	2	0
26 . Extérieur	180						

BRUIT

1/ Isolation Acoustique Ext.Int. peut produit du bruit

2/ Bruit faible

3/ Conversation courante

4/ Source de bruit ou non génée par une source de bruit puissante extérieure.

PROPRETE

1/ Sol mouillé

2/ Sol sec

3/ Moyen

4/ Propre

ACCES EXTERIEUR

0/ Direct

1/ Indirect

2/ Nul

FIG. IV-5 Propriétés des activités du centre socio-culturel

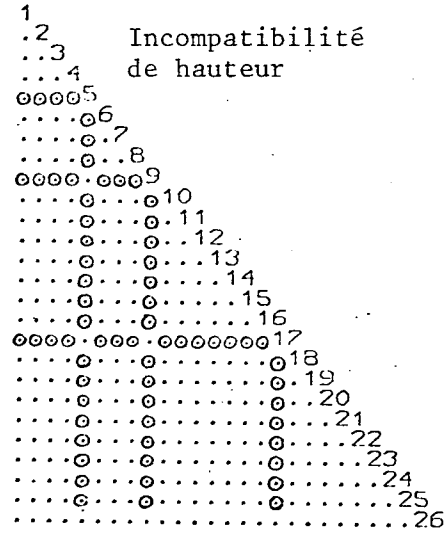
Le but du travail de conception est de dégager une solution qui satisfasse au mieux tout ce qui a été explicité, en respectant en plus les contraintes économiques et technologiques. Pour cela il faut synthétiser les informations précédentes, établir leur interaction et leur priorité.

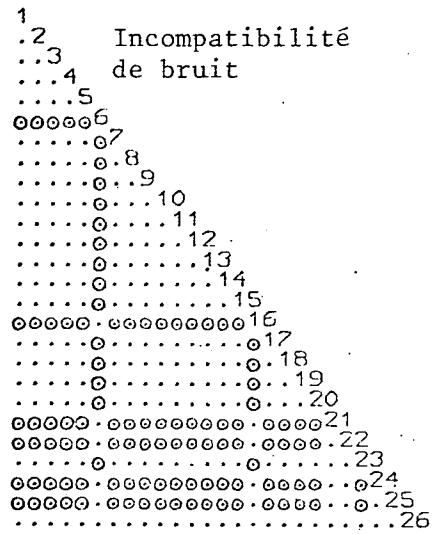
Pour cela on procède à l'analyse des données pour assurer leur compatibilité pour pouvoir les structurer voire les hiérarchiser.

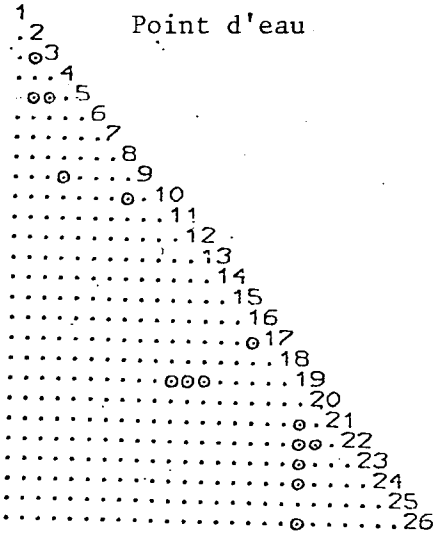
On construit une relation, résultant des relations précédentes. On définit la méthode de prise en compte de différents renseignements et on obtient la relation suivante (voir Fig.IV-6).

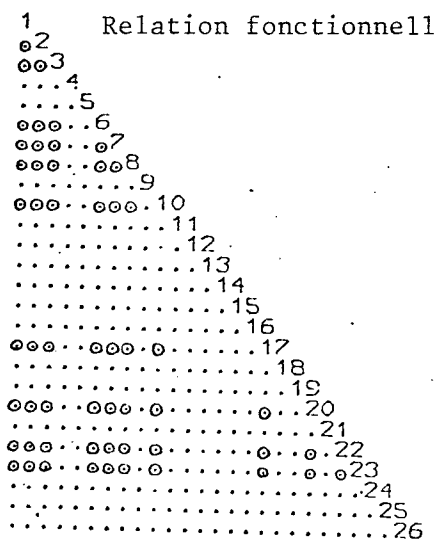
Puis on essaie d'effectuer le répartition dans l'espace (allocation spatiale); le nombre d'objets étant trop important (Fig. IV-7), on peut constater que la solution obtenue n'est pas satisfaisante et il est donc préférable de traiter ce problème par parties.

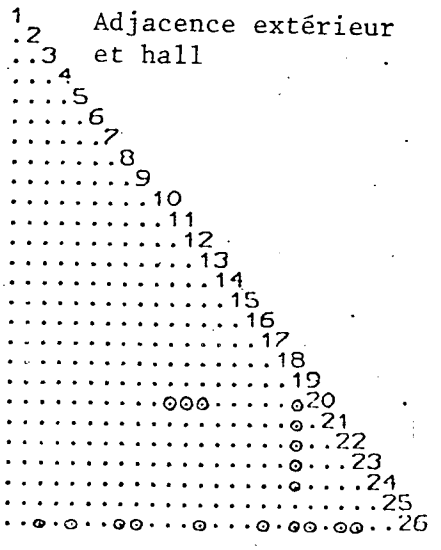
Pour cela on effectue la hiérarchisation du problème, ce qui permet d'obtenir des groupements de relations semblables. Malheureusement le programme (29) avec lequel nous avons traité ce problème, n'accepte qu'une matrice binaire. Nous avons donc applati la matrice résultante en fixant le seuil à cinq. Le résultat est représenté sur la Figure IV-8.











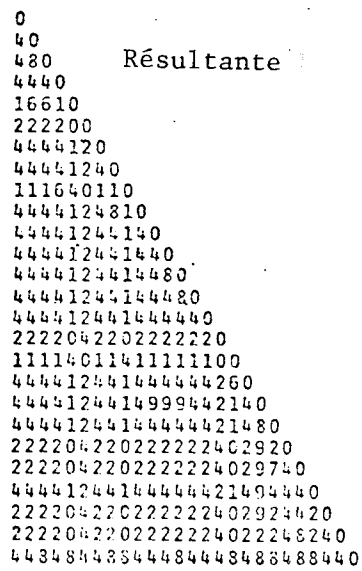


FIG. IV-6 Relation résultante obtenue par superposition de différentes relations

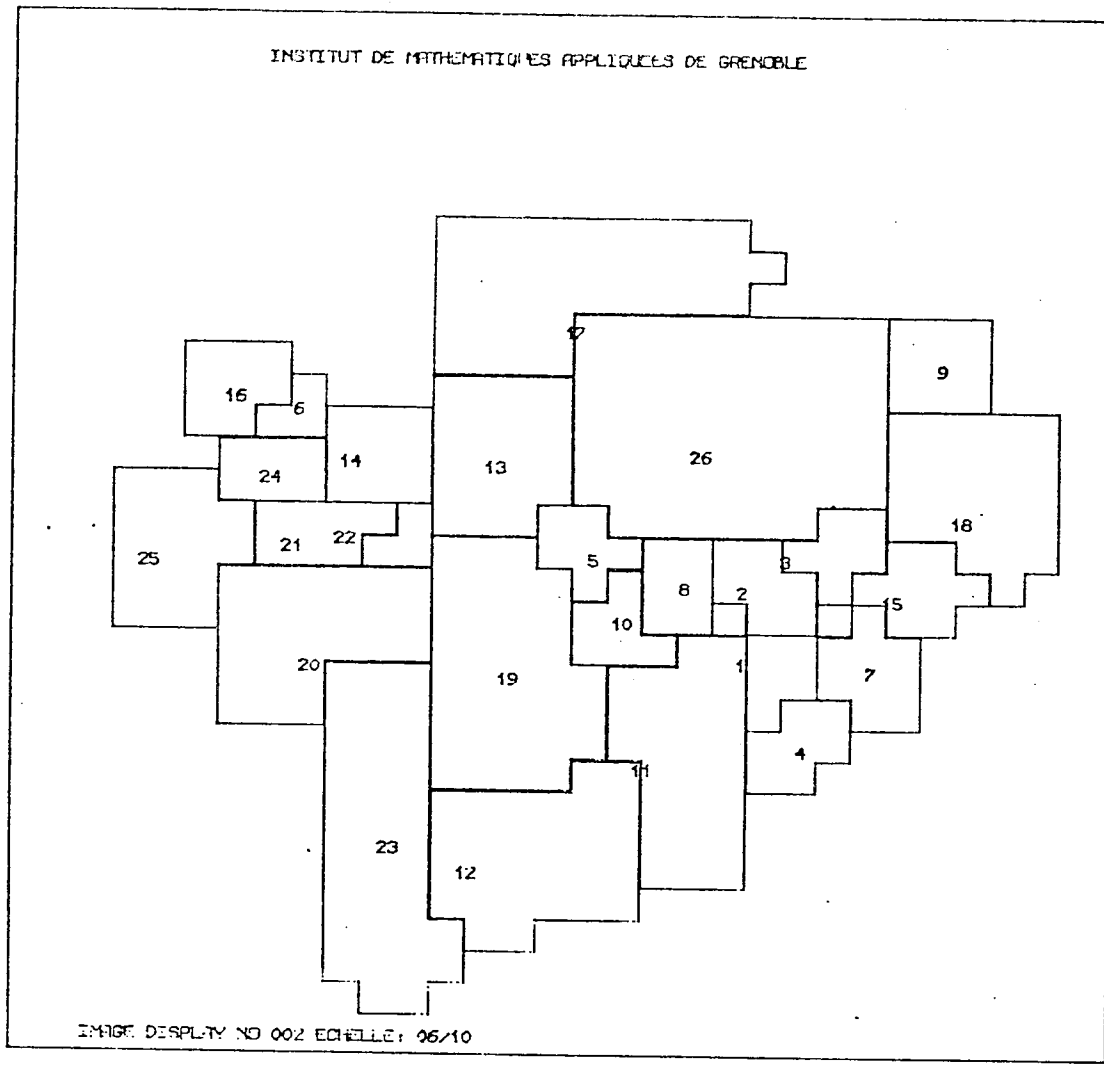


FIG. IV-7-a Répartition spatiale du centre

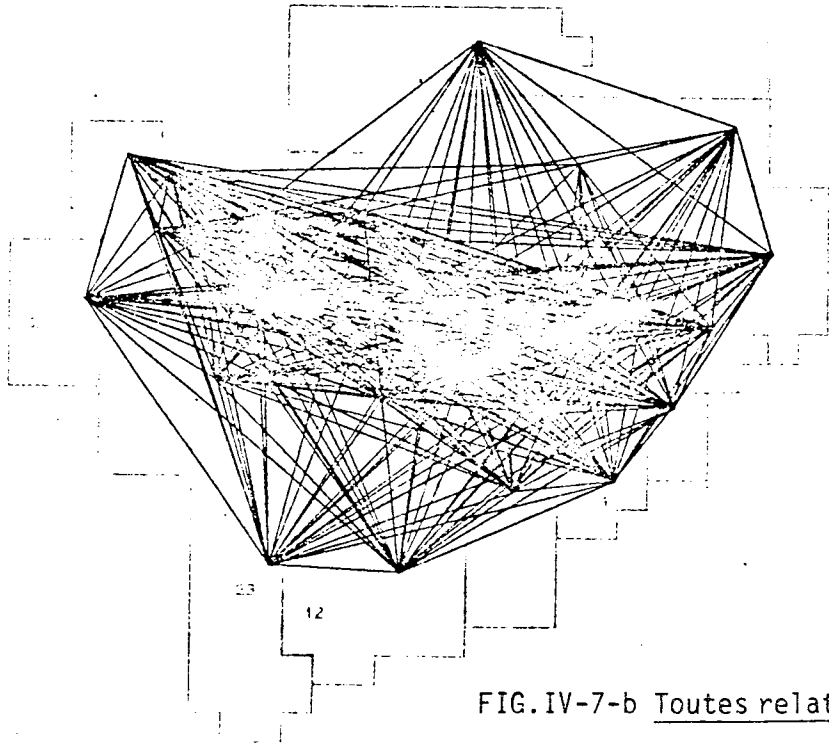


FIG.IV-7-b Toutes relations entre activit

TABLEAU INITIAL

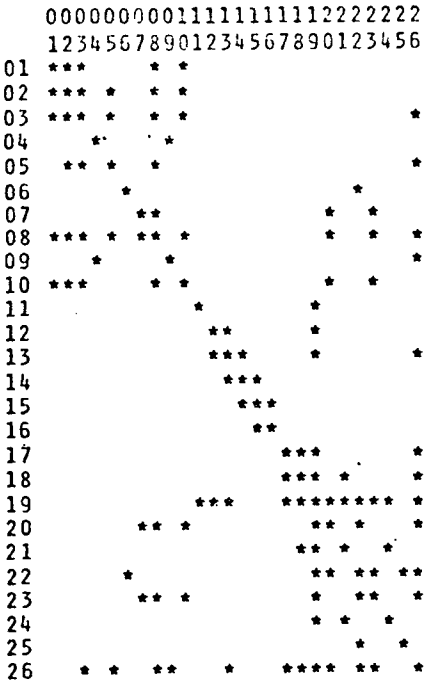
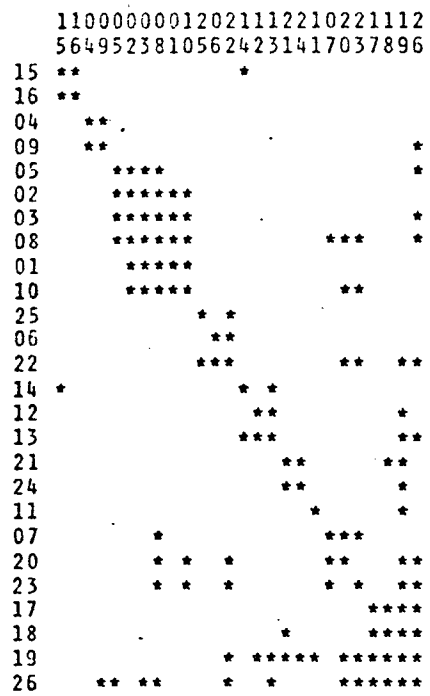


TABLEAU ORDONNE



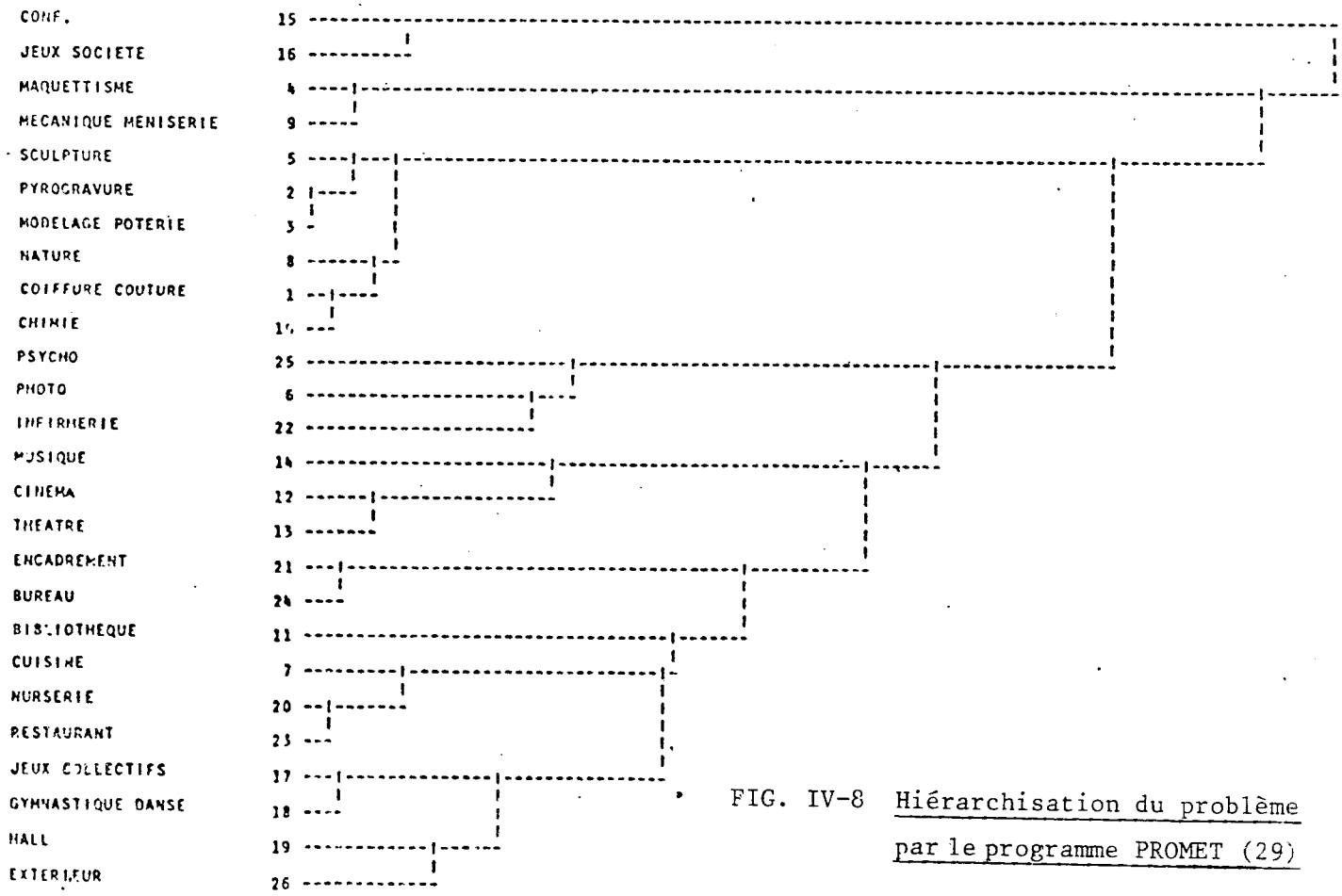


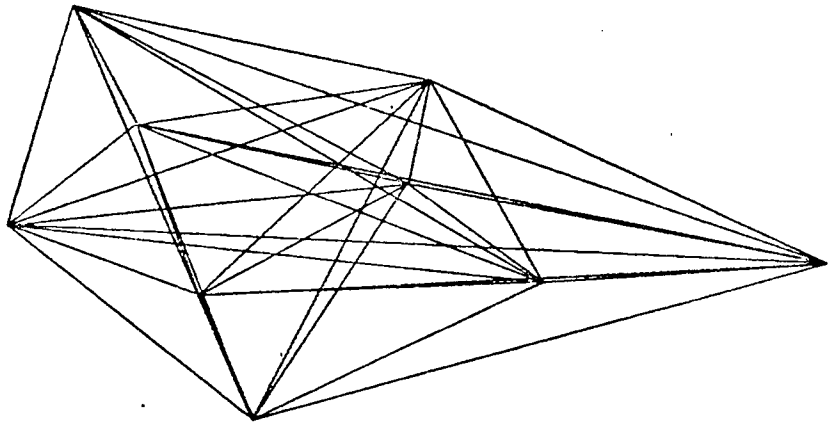
FIG. IV-8 Hiérarchisation du problème par le programme PROMET (29)

Nous pouvons voir qu'un groupement d'activités apparaît et que, pour la suite, nous pouvons soit constituer deux autres groupements moins explicites et traiter à la fin la composition de ces trois groupements, soit travailler avec le noyau ainsi constitué et par la suite faire intervenir un par un ou par petits groupes les activités restantes pour obtenir directement une solution totale.

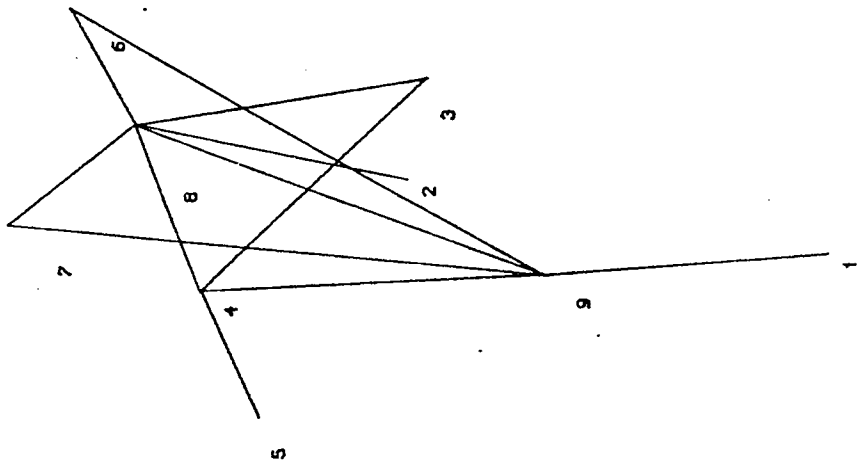
Après avoir dégagé un groupement, nous procédons à l'allocation spatiale c'est-à-dire que l'on produit une répartition spatiale en respectant la surface (éventuellement la forme) et, dans la mesure du possible, les relations indiquées dans la matrice résultante entre activités.

Pour juger la solution obtenue (Fig.IV-9) nous pouvons: d'une part étudier la satisfaction de différentes relations et apporter les modifications éventuelles conversationnellement, d'autre part apprécier la solution en trois dimensions (Fig.IV-10).

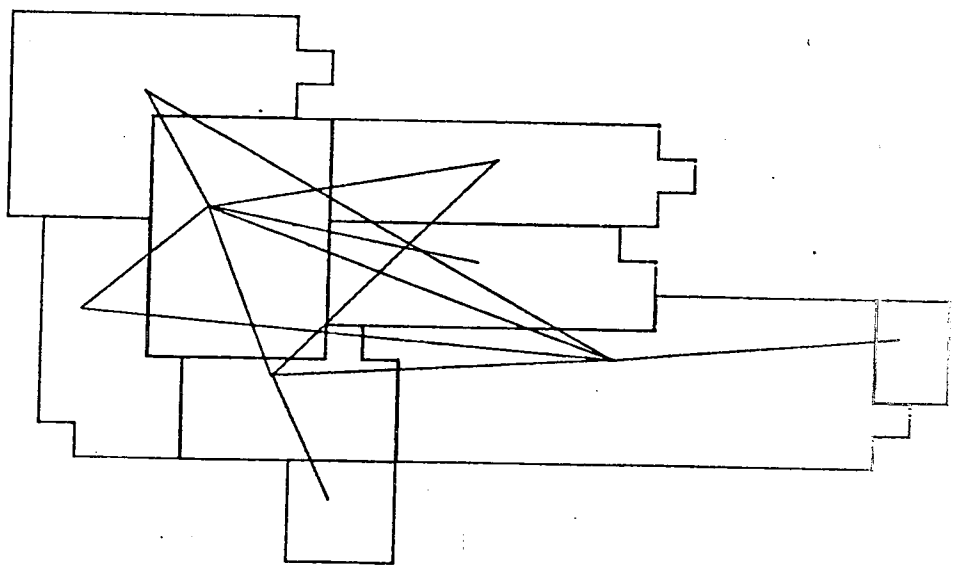
Puis on travaille à l'obtention d'une forme finie. Pour cela on améliore la forme, éventuellement on la modifie pour pouvoir utiliser un système constructif, puis on spécifie les équipements. Voir IV.4, car cette partie n'est pas encore incorporée au système.



Ensemble de relations



Graphe de relations ≥ 7



Répartition spatiale avec la superposition du graphe

FIG. IV-9 Jugement de la solution pour un groupement

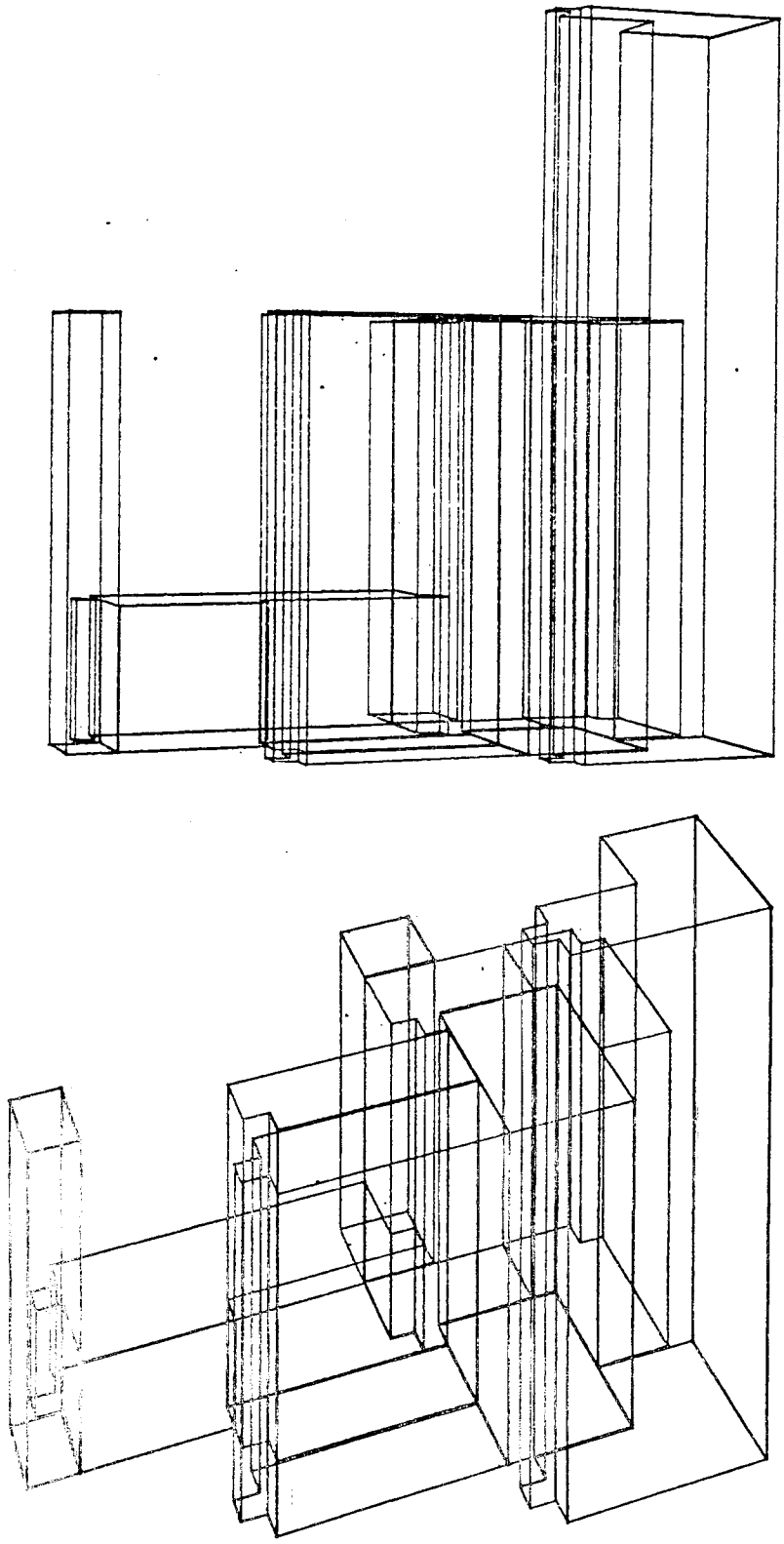


FIG. IV-10 Représentation en 3D de la solution

IV.3.3 Schéma fonctionnel du système

Présentons maintenant plus précisément l'aspect fonctionnel du système SIGMA-Archi. Le schéma complet vu par un informaticien est présenté sur la figure IV-11.

La partie saisie et manipulation d'informations fonctionnelles a pour but d'obtenir et de structurer des informations.

Les informations sont d'une part des unités de conception avec leurs propriétés et d'autre part les relations les liant. Pour des raisons de commodité d'expression et de traitement, nous faisons la distinction entre les relations renseignées et binaires. Les différents traitements permettent les opérations de composition de relations, l'analyse de compatibilité des propriétés et les différentes techniques d'analyse des données.

Dans la phase concernant la production de formes brutes, on procède à la construction, soit manuellement (EDITION), soit par des méthodes de GENERATION (la forme est définie); soit par des méthodes d'ALLOCATION spatiale (on tient compte uniquement de la surface), soit à l'aide d'un langage de génération de formes géométriques indépendament de l'aspect fonctionnel.

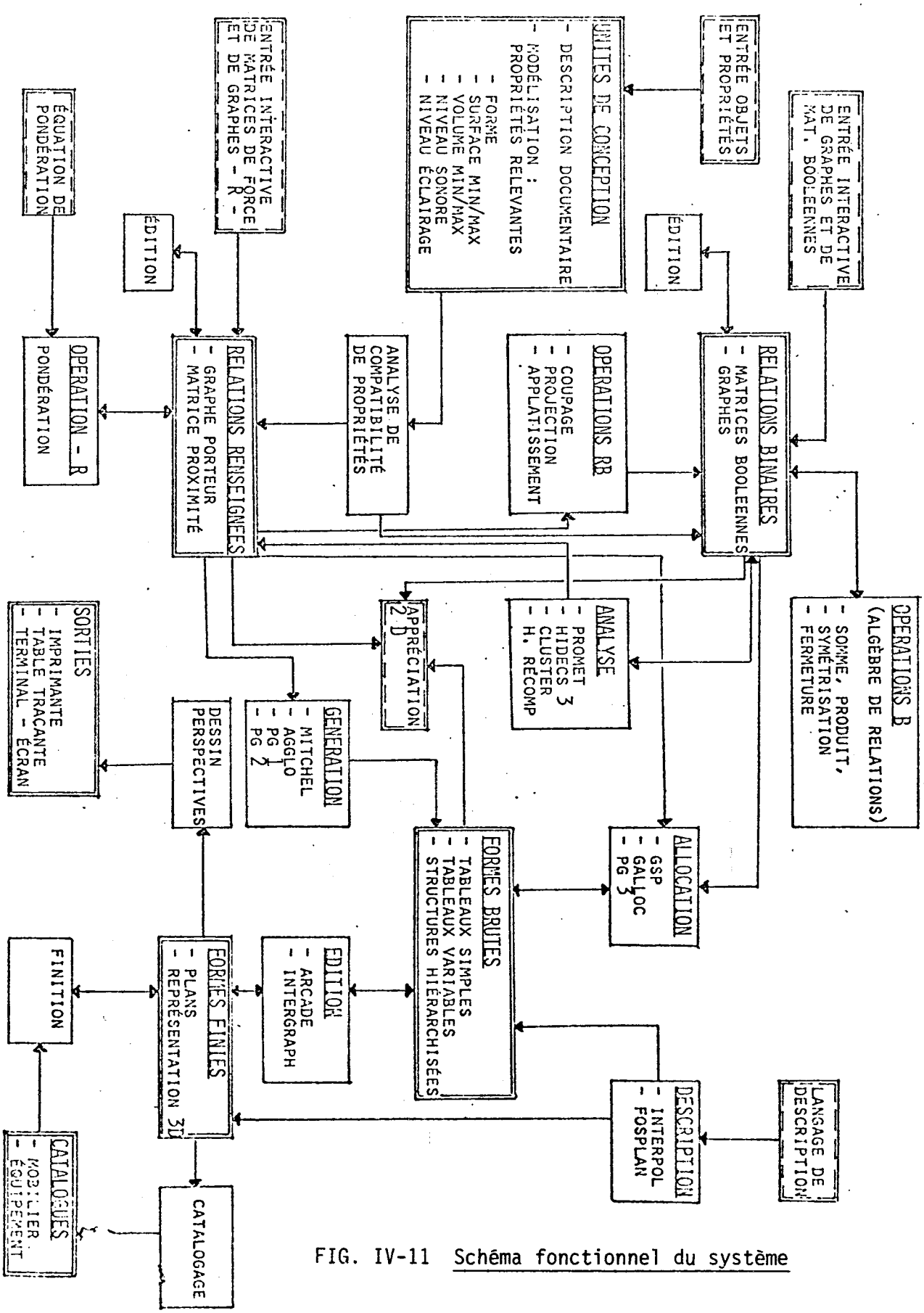


FIG. IV-11 Schéma fonctionnel du système

Dans la phase concernant les formes finies, le langage de génération de formes peut aussi intervenir, ainsi que les programmes d'édition de formes. En plus tout l'aspect technique, les différents catalogues techniques et les méthodes pour les faire intervenir dans le projet étudié (placement d'équipement, réalisation à l'aide d'un système de préfabrication, etc...).

L'aspect appréciation est très important: il est fondamental de disposer d'outils puissants et efficaces qui fournissent à l'utilisateur des renseignements utiles et sous une forme directement exploitable (Fig.IV-12).

IV.3.4 Les MSR comme support formel

Les MSR peuvent recevoir tous ces types d'information et les traiter, donc peuvent servir de support. Nous pouvons constater que les traitements pour chaque phase peuvent être réalisés à partir des primitives MSR.

Analyse de données

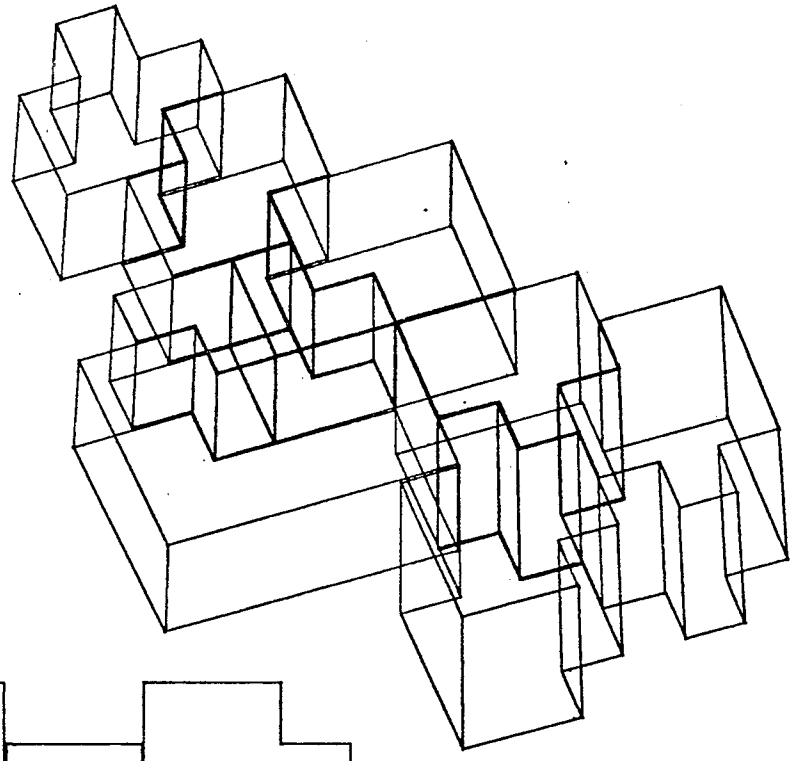
On recherche des ressemblances et on classifie: on travaille sur le MSR de deux manières, dont les résultats sont:

-- une nouvelle hiérarchie de sous-MSR isomorphe à la précédente; elle est obtenue par restructuration du MSR.

Taille		7	3	9	9	8	7	3	6	17
Hauteur	3	2	3	3	2	2	2	2	2	

```
0
20
600
4210
24020
240240
2602440
24024840
240244840
```

```
027282930313233343536373839404 1
026 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
027 0 0 0 3 3 3 0 0 0 0 0 0 0 0 0
028 0 0 3 3 3 3 0 0 0 0 0 0 0 0 0
029 0 0 0 3 3 1 1 0 0 0 0 0 0 0 0
030 0 0 4 4 1 1 1 5 5 5 0 0 0 0 0
031 0 4 4 4 4 1 1 5 5 5 0 0 0 0 0
032 0 4 4 4 9 9 9 2 5 5 6 6 0 0 0
033 0 0 0 0 9 9 9 2 2 6 6 6 8 0 0
034 0 0 0 0 9 9 9 7 7 6 6 8 8 8 0
035 0 0 0 0 9 9 9 9 7 0 0 8 8 0 0
036 0 0 0 0 9 9 9 9 0 0 0 0 0 0 0
037 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



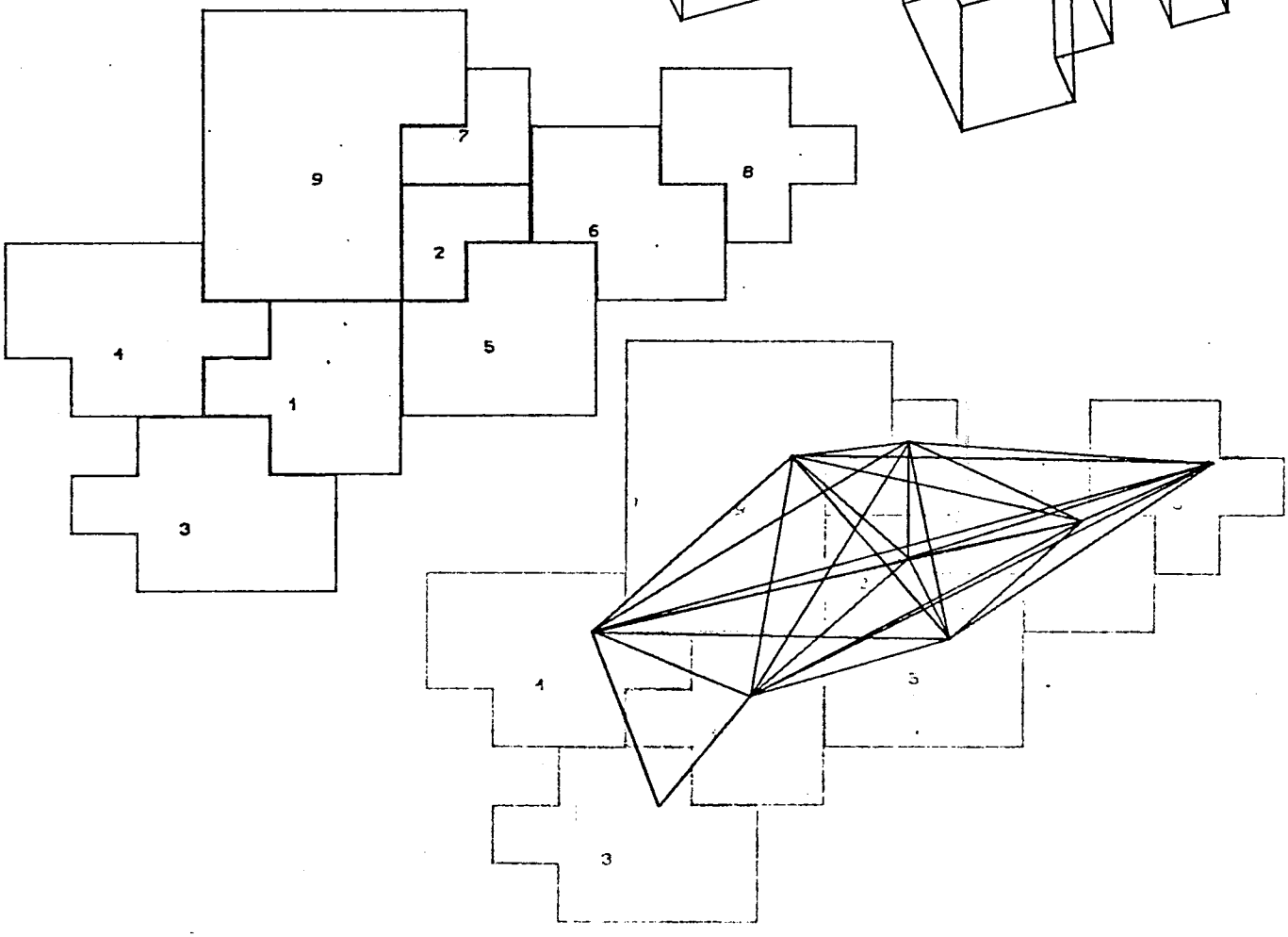


FIG. IV-12 Différentes présentations de mêmes données

-- un nouveau MSR plus renseigné que le précédent; il est obtenu par introduction de nouvelles couches dans le MR; les nouveaux renseignements (relations et propriétés) sont déduits du MR lors de l'analyse (degré de ressemblance, proximité, etc...).

La première méthode a toute sa signification quand le résultat de l'analyse de données est effectivement une hiérarchie. Cependant, d'une part elle ne représente qu'un seul critère de classification, d'autre part elle n'est pas adéquate quand l'analyse de données ne produit pas de hiérarchie stricte, mais une classification avec recouvrements. Ces deux problèmes sont éliminés dans la deuxième méthode où une nouvelle couche est introduite pour chaque critère de classification.

Les algorithmes de classification doivent travailler sur le MBB pour n'avoir à leur disposition que des informations au même niveau et donc ne pas avoir des similarités cachées par les entités non terminales.

Les opérations d'interrogation sont utilisées dans un but de localisation de groupements répondant à un certain critère. Ces critères peuvent concerner tant les propriétés des entités (similarités de propriétés), que les relations (degré de connection).

Les opérations structurelles et de modification servent à former le nouveau MSR.

Allocation spatiale

L'allocation spatiale travaille sur le MSR pour obtenir une première répartition dans l'espace des éléments représentés par les entités; elle doit respecter au mieux les propriétés, et toutes les contraintes liant les entités.

Selon le type de problème, forme est soit définie dès le départ, soit générée à partir des propriétés, soit générée par l'algorithme de placement lui-même.

Le choix de l'entité à placer peut être réalisé par succession d'opérations d'interrogation sur le MSR.

Les fonctions de placement utilisent surtout les opérations de modification. Le placement consiste non seulement en l'affectation d'une position (et éventuellement génération de forme) mais en plus en l'indication dans le MSR des conditions dans lesquelles ce placement s'est effectué (c'est-à-dire que l'on indique les contraintes satisfaites et non satisfaites, les valeurs d'informations au moment du placement comme distance, surface, etc...).

Spécifications détaillées

L'arrangement géométrique définitif est basé sur le résultat des ases précédentes. L'introduction de solutions techniques sert à la térialisation de la solution retenue. Il faut aussi tenir compte des

informations introduites précédemment dans le MSR comme les choix techniques imposés.

En plus on essaie de résoudre par des solutions techniques les conflits dûs aux contraintes non satisfaites.

La spécification des équipements utilisés termine cette phase.

La structure MSR permet d'avoir constamment à sa disposition toutes les données du problème. Les fonctions d'interrogation et de modification servent à son actualisation.

La continuité est donc assurée du MSR essentiellement relationnel jusqu'au MSR essentiellement topologique et technique.

On couvre toutes les phases de la conception à l'aide d'un seul formalisme ce qui représente un apport considérable et permet une meilleure maîtrise de la conception.

IV.4 Modélisation de plans

Ce qui suit concerne la phase de spécification définitive. Le lien avec le MSR n'apparaitera pas explicitement car nous pensons que son utilité a été demontré et que le lecteur pourrait lui même faire ce lien. Nous nous consacrerons entierement au problème lui même.

Après avoir obtenu une première répartition de volume, il faut pour aboutir à un prototype réalisable, compact et plus esthétique effectuer un certain nombre de modifications: améliorer la forme, préciser les solutions techniques non imposées, éventuellement faire intervenir des équipements.

On réalise ainsi des plans sous forme habituelle.

IV.4.1. Description générale

Nous avons développé ce programme en envisageant deux possibilités d'utilisation: la première que nous avons évoquée plus haut: c'est le cas d'une conception complètement assistée; mais les discussions avec les architectes nous ont permis de constater qu'une nécessité d'aide se fait actuellement sentir dans la conception architecturale classique, et au niveau des dessins, et pour l'évaluation et l'appréciation d'une proposition. C'est pourquoi nous avons inclus une deuxième possibilité: utilisation séparée du système qui fournit un support d'information actif depuis "l'esquisse"

jusqu'aux "plans de construction": cela nécessite la création directe d'une proposition initiale remplaçant les étapes préalables.

Après création du schéma initial l'utilisateur dispose, pour la phase suivante (principale), de fonctions permettant de juger la qualité de son projet, éventuellement de modifier ou d'ajouter aussi bien des données géométriques que techniques. Quand il obtient une solution satisfaisante (ou tout au moins intéressante) il peut la sauvegarder, la sortir sur la table traçante, imprimer la description correspondante.

IV.4.2 Structure de données

L'élaboration de la structure interne a été guidée par:

-- l'observation d'un certain nombre de plans de construction, et un recensement des informations nécessaires à l'architecte au cours de son travail.

-- le désir de la rendre compatible avec un environnement conversationnel où l'on peut construire ou modifier cette structure.

-- utilisable à d'autres fins telles que calculs d'aires, de résistances, de coûts, etc...

Nous avons retenu les hypothèses de travail suivantes:

-- un plan est un ensemble de pièces, pour autant qu'on y ajoute le pourtour initial considéré comme pièce spéciale à l'intérieur de laquelle on peut en créer d'autres.

-- une pièce est formée d'une suite de murs orthogonaux qui se ferment (les pans coupés et les formes géométriques autres ne formant que de rares exceptions).

Sur un plan ainsi défini on fait apparaître les solutions techniques et des "accessoires" possédant des caractéristiques, que l'on peut essayer de classifier:

-- caractéristiques générales:

. Epaisseur des murs
. Constituant de ces murs
. Hauteur sous plafond
. Revêtement et constitution du sol ou du plafond

-- accessoires: portes, fenêtres de caractéristiques suivantes:

. Type dans la normalisation en usage
. Situation
. Hauteur au dessus du sol

-- enfin apparaît une dernière catégorie d'éléments qui regroupe:

. Conduits et canalisations

. Accessoires d'ameublement et appareils sanitaires

La structure hiérarchisée est la suivante:

1) Etude/ensemble de projets

2) Projet/ensemble de plans

3) Plan/ensemble de groupement de pièces

4) Groupement de pièces/ensemble de pièces

4) Pièce/ensemble de murs + accessoires figurant dans le pièce

6) Mur/mur + accessoires portés par ce mur

Le point 1) correspond à la notion d'ensemble d'immeubles.

Le point 2) correspond à la notion d'étage.

Il est à noter que dès ce stade, il faut relever des contraintes existant entre deux plans d'un même projet (canalisation, cages d'escaliers, murs porteurs, etc...).

La notion de groupement de pièces a été introduite non seulement pour faciliter le travail (on doit tenir compte de la taille de l'écran et de la capacité de la mémoire) mais encore pour permettre de définir des sous-ensembles cohérents remplissant une certaine fonction comme appartement, partie "repos" d'un appartement, etc...; ces exemples montrent l'intérêt d'introduire la récursivité dans cette

notion.

Pour séparer nettement l'aspect géométrique de l'aspect technique nous avons organisé notre structure de la façon suivante: d'une part une structure comprenant les données géométriques, d'autre part un catalogue de toutes les caractéristiques techniques (constituants, revêtements de sol, accessoires, etc...), la liaison entre eux se faisant dans les deux sens. L'utilisation du catalogue empêchera une multiplication exagérée des solutions techniques. Il pourra être modifié directement par des fonctions de création, d'interrogation et de mise à jour spécifiques. A chaque instant de la conception on peut connaître:

-- tous les endroits où une solution donnée est utilisée.

-- la liste de toutes les solutions utilisées.

-- à une échelle encore plus grande, des statistiques sur l'utilisation des diverses solutions pendant l'élaboration de différents projets.

La structure interne est illustrée par la fig. IV-13.

Les fonctions de modification dans un plan (translation de murs, de pièces, etc...) font intervenir un deuxième aspect fondamental, l'aspect relationnel.

L'agrandissement d'une pièce suppose le rétrécissement d'une ou

plusieurs pièces adjacentes. Cet aspect a été matérialisé par une matrice d'adjacence booléenne donnant les relations entre les pièces (fig. IV-14).

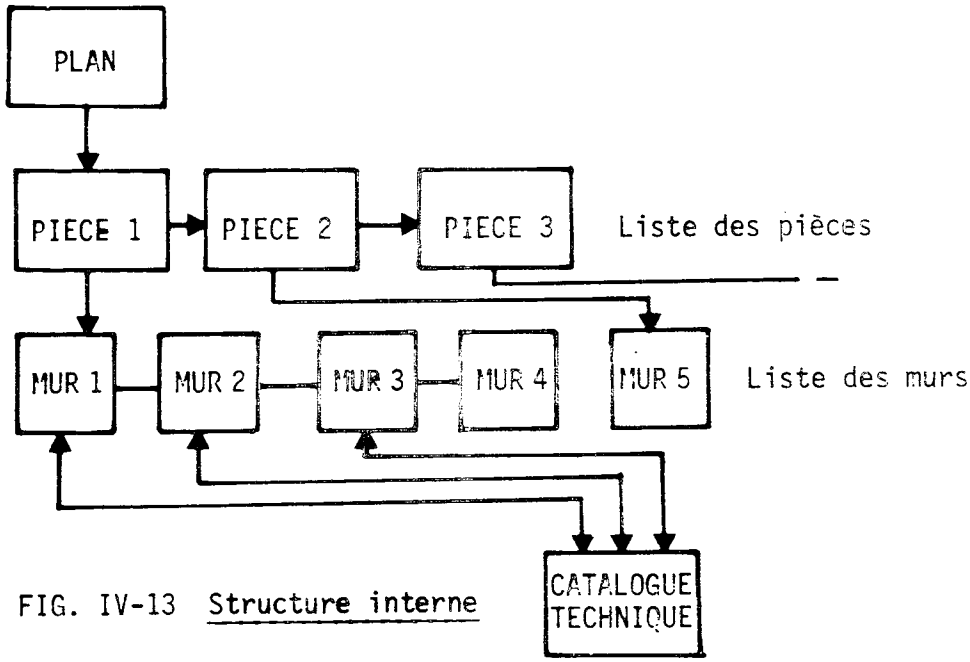


FIG. IV-13 Structure interne

Exemple

la pièce P5 a au moins 1 mur commun avec les pièces P0, P2, P3
la pièce P2 a au moins 1 mur commun avec les pièces P1, P5 et P4
la pièce P4 a au moins 1 mur commun avec les pièces P2 et P3.

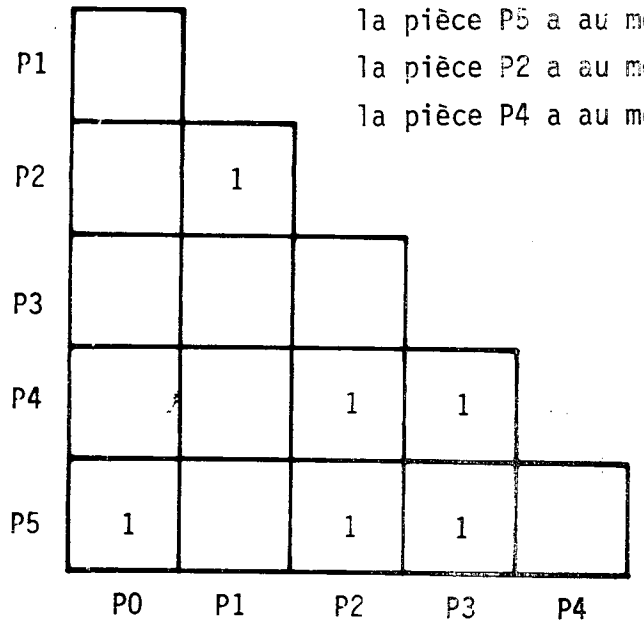


FIG. IV-14 Matrice d'adjacence

IV.4.3 Méthode d'approche

Comme nous considérons la partie géométrique en tant que support de la partie technique, il faut définir la géométrie avant d'aborder les précisions techniques (c'est-à-dire définir la géométrie de la pièce avant de préciser les équipements, etc...).

Pour la partie géométrique l'utilisateur peut procéder en trois phases:

- modélisation
- analyse
- sortie

Disposant de fonctions prédéfinies, que nous verrons par la suite, il peut dans la première phase créer ou modifier la répartition géométrique de son application. Dans la deuxième il peut, à l'aide d'autres fonctions, interroger et analyser le projet afin de pouvoir l'apprécier. S'il est satisfait, il peut sortir et passer dans la deuxième étape et cela avec ou sans sauvegarde de résultats; si par contre il veut modifier certaines choses il revient dans la première phase (modélisation).

Nous travaillons avec les traits, et faisons la distinction entre les différents composants par fonctions (création de mur, création de pièce, etc...). Par contre pour la partie technique nous ne pouvons agir de même à cause de la grande variété des informations à représenter.

En conséquence, la structure de données est complétée par un catalogue et la méthode d'approche par une phase supplémentaire conduisant à:

1) définition
2) modélisation
3) analyse
4) sortie

La phase de définition correspond au travail sur le catalogue; à la deuxième phase vient s'ajouter l'indication des liens entre support géométrique et catalogue; les deux autres restent inchangées.

IV.4.4 Traitements des données géométriques

L'analyse des différentes opérations que pouvait effectuer un architecte sur un plan nous a conduit à retenir les fonctions suivantes:

- Entrée d'un plan initial (contour ou plan inachevé)
- Création de pièces
- Création de murs
- Suppression de pièces
- Suppression de murs
- Translation de pièces
- Translation de murs

Plan initial: le plan initial sera soit un pourtour initial composé d'une suite de murs orthogonaux qui délimitent l'enceinte initiale, soit un ancien plan inachevé.

Les murs sont rentrés les uns à la suite des autres à l'aide du photostyle sur l'écran du terminal graphique, et successivement rangés dans la structure pour former le plan initial.

Remarque: les architectes travaillant, dans 80% des cas, sur des pourtours initiaux rectangulaires, nous avons prévu l'entrée simplifiée du rectangle.

Génération de murs: entrée au photostyle (PO) d'un point représentant une extrémité du mur, puis désignation d'une direction de génération, parmi 4 possibilités: NORD, SUD, EST, OUEST.

Génération de pièces: entrée au PO d'un point indiquant l'emplacement de l'angle de la pièce, puis désignation d'une direction de génération parmi 4 possibilités: NORD-OUEST, NORD-EST, SUD-OUEST, SUD-EST.

Génération de pièces dites "en verrue" (fig. IV-15): cette fonction qui représente une synthèse des deux dernières, a été réalisée à la demande des architectes. Entrée au PC de deux points figurant les deux angles extérieurs de la pièce, puis désignation d'une direction parmi 4 possibilités:NORD, SUD, EST, OUEST.

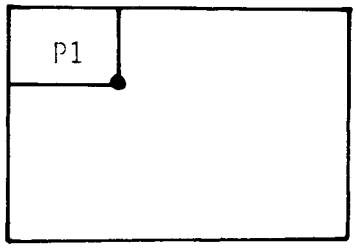
Notion de pièce libre: l'algorithme de génération de pièces a pour but la résolution du problème suivant: étant donné un point et une direction de génération (NO, NE, SO, SE) trouver tous les murs qui définissent le pourtour de la pièce.

La notion de pièce libre a été introduite dans le but d'optimiser cette recherche.

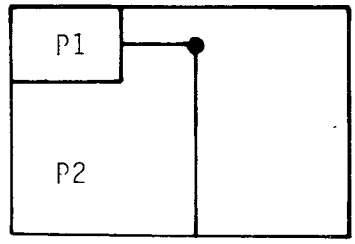
Par définition la pièce libre est l'ensemble des murs du plan pouvant faire partie d'une pièce créée postérieurement, sous réserve que l'on ne crée pas de pièces à l'intérieur d'une pièce.

Cette notion sera facilement comprise à l'aide de la fig. IV-16 (a, b, c, d).

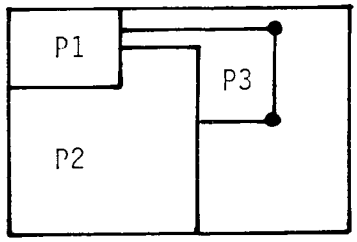
Cette pièce libre est conservée dans une zone mémoire complètement indépendante de la structure de données elle-même.



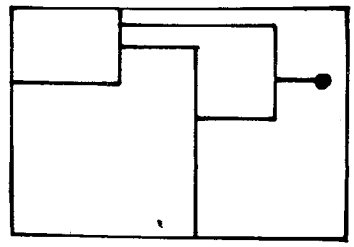
a) Pièce direction : N.O.



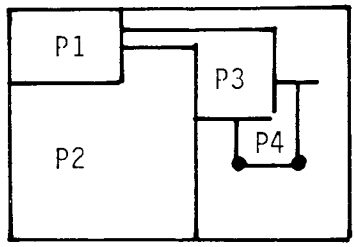
b) Pièce direction : S.O.



c) Pièce en verrue direct.O.



d) Mur-Direction : Ouest



e) Pièce en verrue direction : Sud

FIG. IV-15 Différentes fonctions géométriques

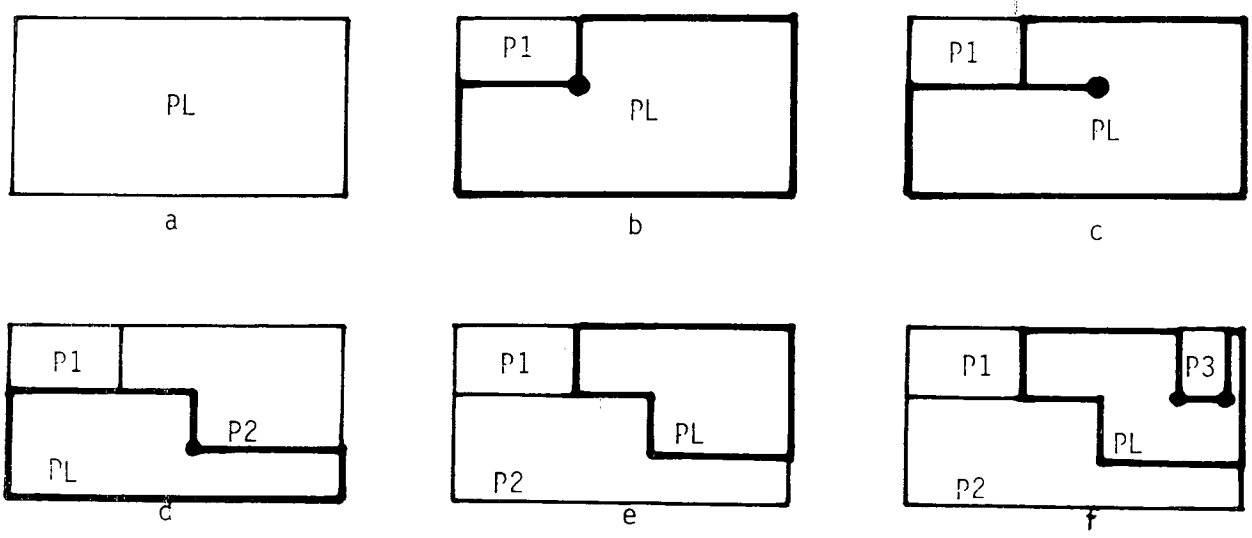


FIG. IV-16 Evolution de la Pièce Libre

On voit que puisqu'on ne s'intéresse qu'aux murs de la pièce libre:

-- lors de la recherche des murs d'appui, le nombre de murs à examiner est considérablement réduit.

-- la recherche des murs constituant la pièce consiste en l'itération de l'algorithme suivant: chercher un mur orthogonal au dernier mur trouvé, et possédant une extrémité commune avec ce dernier.

Nouvelle pièce libre: les créations n'étant possibles que dans la pièce libre, pour que l'utilisateur puisse créer dans une pièce déjà existante, on a introduit la notion de changement de la pièce libre, bien visible sur la fig. IV-16(d, e,f). De plus, à la fin de la phase de création, l'utilisateur peut, s'il le désire, donner un nom à la pièce libre.

des fonctions suivantes:

- Suppression de murs seuls
- Suppression de pièces
- Agrandissement de la pièce libre
- Agrandissement d'une autre pièce

Elles font intervenir, à l'aide d'algorithmes assez complexes, des changements dans la structure géométrique, dans la pièce libre,

dans la matrice d'adjacence. La translation de pièce s'effectue à l'aide des translations successives des murs la constituant. Lors de la suppression d'une pièce, on doit indiquer si la place obtenue doit être rendue à la pièce libre ou s'il s'agit d'union avec une autre pièce.

Ces outils ont été testés par les architectes, en vue de déterminer s'ils étaient suffisants pour le tracé de n'importe quel plan. Ils ont remarqué que le travail en proportion, par rapport au contour et aux pièces déjà créés, peut être gênant, même si on fait apparaître l'échelle sous forme de grille (avec possibilité d'initialisation) ; en plus ce mode de travail n'est accepté que par certains architectes. C'est pourquoi nous avons implémenté dans les fonctions une option permettant l'entrée de longueurs précises.

Nous introduisons également des fonctions d'ajustement pour faciliter l'alignement de murs.

IV.4.5 Aspect technique

Comme nous l'avons vu précédemment la partie technique nécessite un câtaloque contenant les solutions proposées; nous pouvons décrire l'organisation globale du cataloque de la façon suivante:

Nom générique . Nom symbolique . Caractéristiques

C'est une structure arborescente dans laquelle:

-- nom générique précise le type d'accessoire, de solution technique ou d'équipement (porte, fenêtre, mur en béton, lit, etc...).

-- nom symbolique (optionnel) facilite la désignation des solutions courantes (pour éviter de préciser les caractéristiques à chaque utilisation).

-- caractéristique: elle contient toutes les données nécessaires, stockées d'une façon la plus commode et la plus souple possible (pour une fenêtre par exemple: le type , les dimensions, le nombre de battants, etc...).

On peut constater qu'au nom générique il faut associer une représentation graphique (donc les noms génériques doivent en principe être prévus à l'initialisation du système; de plus, certaines caractéristiques pouvant influencer la représentation graphique (le nombre de battants d'une porte par exemple), il faut les spécifier précisément et comme telles. Notons en passant qu'on ne pourra pas en général représenter les sclutions d'une façon habituelle mais seulement schématiquement; on choisira une représentation la plus simple possible mais suffisamment claire pour être précise.

Nous pensons donner à chaque utilisateur un cataloque personnel, plutôt que d'utiliser un cataloque global contenant un grand nombre de

solutions possibles. Les fonctions de création, d'interrogation et de mise à jour lui permettront de constituer un catalogue "sur mesure" ce qui réduira considérablement sa taille et rendra son utilisation plus pratique (appel par nom symbolique).

Pour incorporer ces solutions dans le support géométrique, les fonctions doivent effectuer:

-- la désignation (pour choisir une solution donnée).

-- le placement (pour indiquer l'endroit où doit être placée cette solution).

-- l'interrogation sur la solution utilisée.

-- l'annulation.

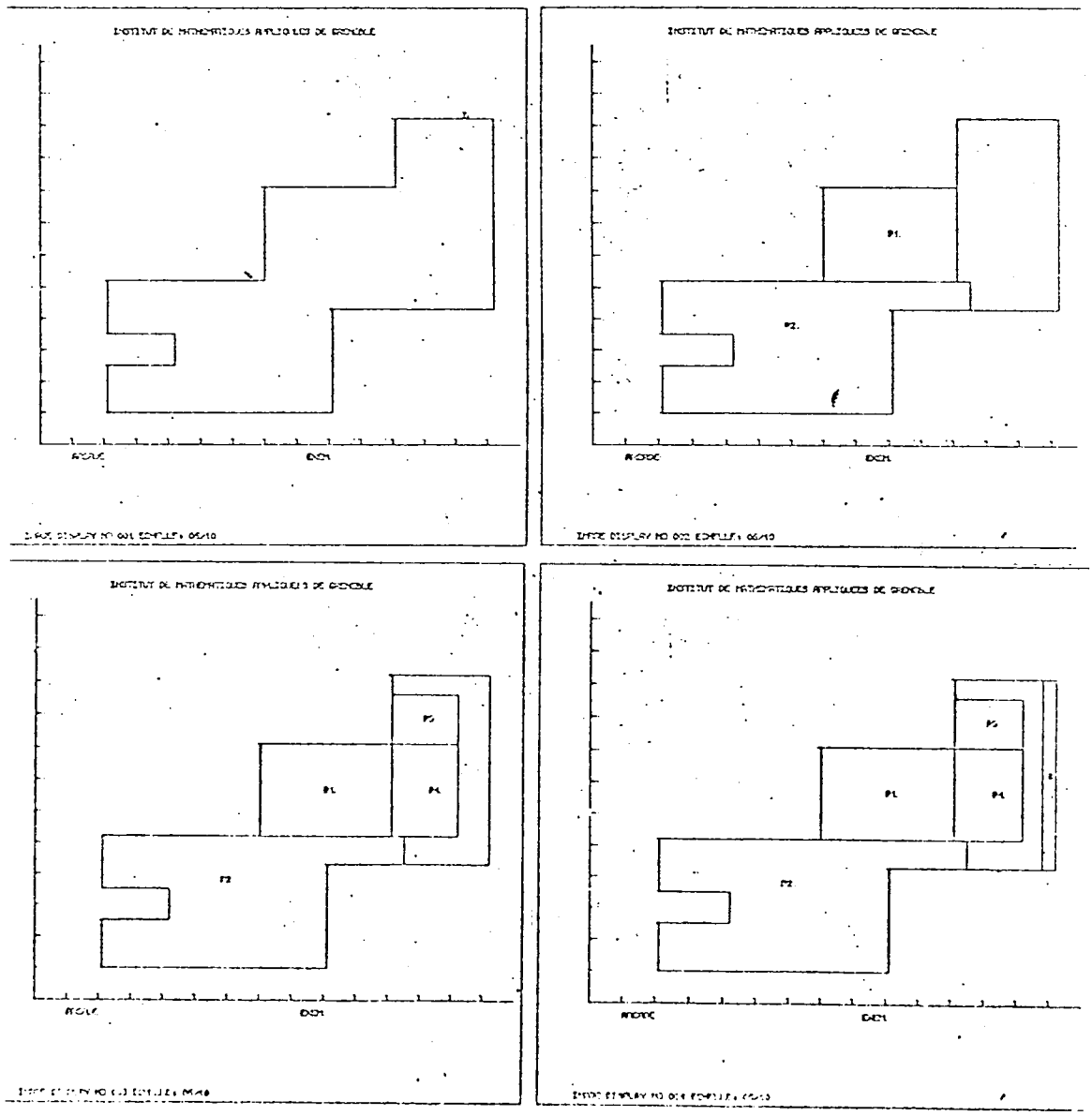
-- la modification (remplacer une solution par une autre).

Nous allons décrire, à titre d'exemple, l'opération de placement: l'appel de la fonction de placement aura pour effet l'affichage de tous les noms génériques disponibles (on peut envisager une sélection dans le cas où l'on en aurait trop). On désigne le nom générique correspondant et comme réponse on obtient la liste des noms symboliques. Si l'utilisateur trouve une proposition satisfaisante il la désigne et il passe dans l'état de placement proprement dit, sinon il demande de passer dans l'environnement de définition; il peut alors

regarder en détail des objets s'y trouvant, en modifier et en ajouter. Quand il a fini il passe dans l'état de placement avec l'objet choisi.

Remarque: on se rend évidement tout de suite compte que la manipulation de solutions techniques peut être effectuée à l'aide du MMS (modèle pour les manipulations symboliques) particularisé.

IV.4.6 Exemple d'évolution d'un plan



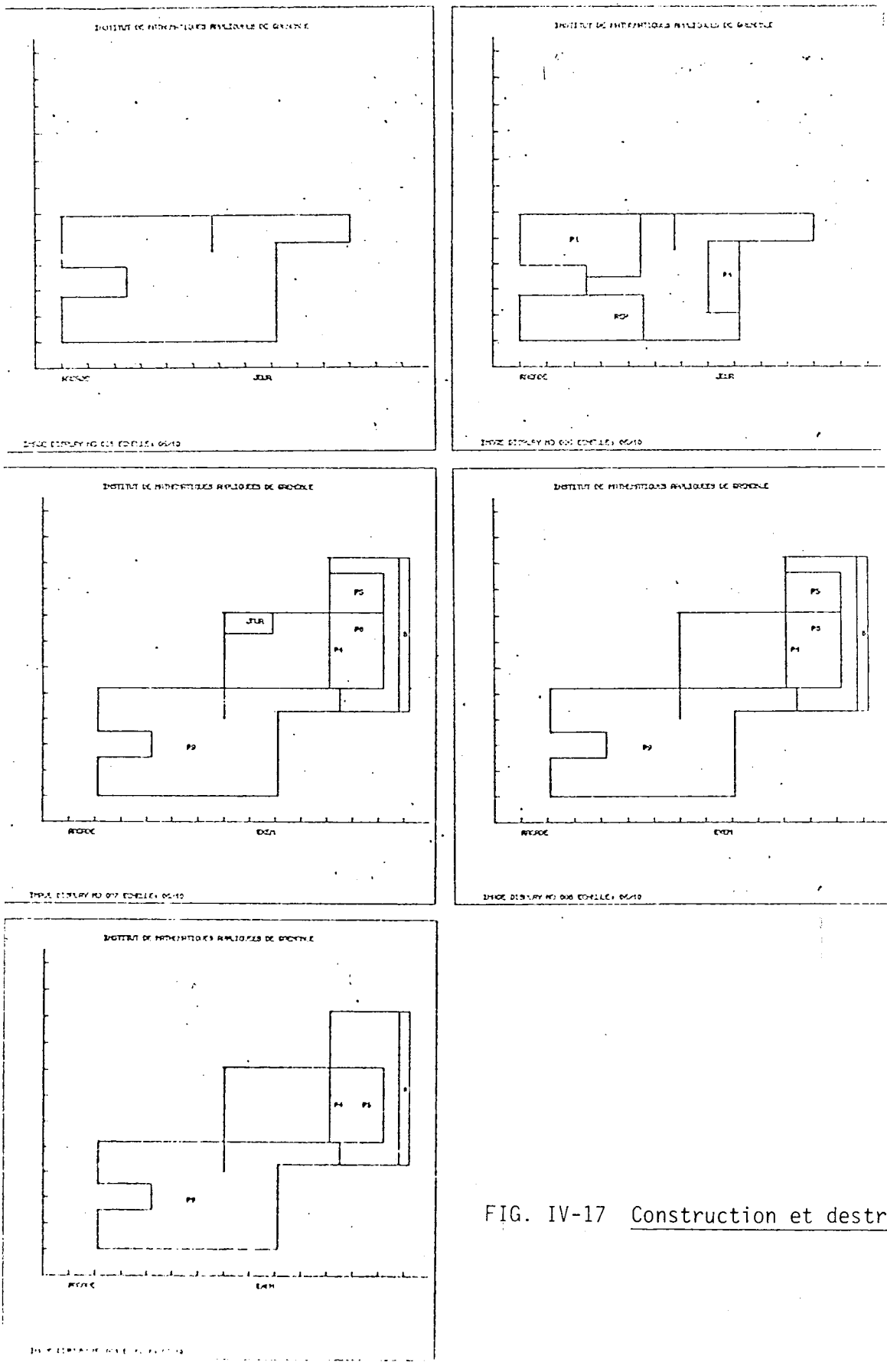


FIG. IV-17 Construction et destruction d'un plan

IV.5 Allocation spatiale

Dans le cadre de nos travaux sur l'architecture nous étudions le problème d'allocation spatiale. C'est RIVERO qui travaille sur ce problème; comme il doit présenter bientôt les résultats obtenus, nous n'aborderons ce problème que succintement, en présentant un programme qui nous paraît intéressant car il correspond à la philosophie du système SIGMA-Archi.

Du point de vue des informations qu'il prend en compte, il n'est pas différent des autres programmes; c'est-à-dire qu'il permet de travailler avec une seule matrice de relations et les objets sont définis soit par leur surface, soit par leur forme.

Le traitement lui-même est déjà intéressant car il prend en compte la forme des objets et, lors du placement, non seulement il tient compte de la relation des objets, mais aussi il effectue une composition optimale. Il peut aussi travailler dans une enveloppe qu'on a définie.

En plus le programme est entièrement interactif et permet à l'utilisateur de suivre les décisions, de les approuver ou les contester, d'en imposer de nouvelles, de modifier la répartition déjà effectuée et d'imposer la partie de l'univers où doit travailler le programme.

Le lien entre l'utilisateur et le programme peut donc être très étroit, ce qui peut faciliter la recherche d'une solution satisfaisante.

IV.6 Changement de représentation

(Exemple de transformation)

Nous présentons ici un exemple de transformation de représentation. Il s'agit de passer d'une représentation sous forme de grille d'occupation à une représentation par contour.

Le problème est donc de trouver le contour de chaque objet et de générer les segments de longueur maximale que l'on utilise par exemple pour une représentation graphique.

Pour cela on balaye le tableau de gauche à droite et de haut en bas et chaque fois que l' on trouve une case non nulle, donc occupée, avec un numéro d'objet non encore traité on applique le processus de transformation suivant.

Le processus de transformation est dirigé par une grammaire. Elle a pour but de déterminer la forme de l'objet et de commander la génération des segments.

L'aspect symetrique du problème permet de minimiser le nombre de cas à considérer. Nous travaillons donc dans les quatre directions avec le même ensemble de règles.

Parmi les neuf règles que nous utilisons il y en a six qui s'appliquent en cours de traitement et et elles sont parametrées par la direction. Les trois restantes interviennent en fin de processus quand on revient à la case de départ.

Le contour de l'objet est parcouru dans le sens trigonométrique. La nature du balayage permet de commencer toujours par la direction SUD(3).

Pour chaque direction on cherche dans les cases voisines la case contenant le même numéro d'objet, dans l'ordre indiqué pour chaque direction sur la figure IV-18. Le numéro d'ordre correspond au numéro de la règle à appliquer.

Dans la règle on génère les segments terminés et on détermine la nouvelle dimension. Les six premières règles pour la direction SUD(3) ainsi que les règles de terminaison se trouvent sur la figure IV-19. Dans le cas général le calcul de la nouvelle direction s'effectue à partir de la direction courante par les opérations données sur la figure. Pour retrouver les directions 1,2,3,4 ce calcul s'effectue modulo quatre plus un. La règle sept s'applique pour une case seule et les règles huit et neuf pour terminer le processus.

Sur la figure IV-20 on présente le déroulement du processus sur petit exemple.

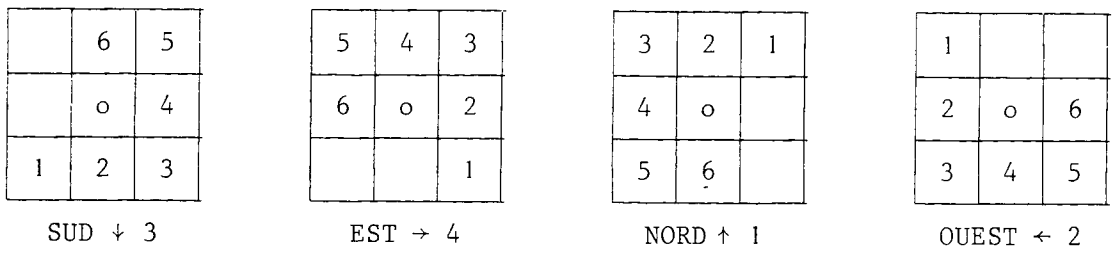


FIG. IV-18 Tableaux dirigeant le choix de régle pour chaque direction.

0	1	2	2
1	1	1	2
3	1	0	4
4	4	4	4

OBJET	N° DE REGLE	DIRECTION	NOUVELLE DIRECTION
1	1	3	2
	5	2	3
	5	3	4
	5	4	1
	9	1	-
2	3	3	3
	6	3	1
	4	1	2
	8	2	-
3	7	3	-
4	1	3	2
	2	2	2
	2	2	2
	6	2	4
	2	4	4
	2	4	4
	4	4	1
	9	1	-

FIG. IV-20 Exemple de déroulement de la transformation

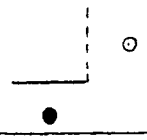
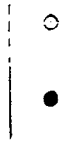
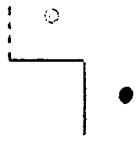
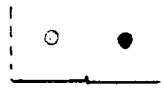
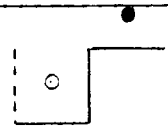
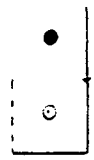
N°	FORME	DIRECTION D	NOUVELLE DIRECTION (ND)	NOMBRE DE SEGMENTS GENERES
1		3 ↓	2 ← ND = D - 1	1
2		3 ↓	3 ↓ ND = 1	0
3		3 ↓	3 ↓ ND = D	2
4		3 ↓	4 → ND = D + 1	1
5		3 ↓	4 → ND = D + 1	3
6		3 ↓	1 ↑ ND = D + 2	2
7		3 ↓		4
8		2 ←		1
		1 ↑		2

FIG. IV-19 Règles de la grammaire

IV.7 CAAO: Principaux centres et manifestations

Dans le domaine de la C.A.O. en architecture les points de recherche importants se situent aux Etats Unis (MIT, Carnegie-Mellon University, University of Utah, UCLA) et en Grande Bretagne (ABACUS, ARU-Edimburg University, CABD Center-Liverpool University, CAD Center et LUBFS Center-Cambridge). La France a été très longtemps en retard, mais il y a un renversement de la tendance grâce, en particulier, aux efforts de l'Institut de l'Environnement.

Depuis 1968 il y a eu de nombreux congrés presque entièrement consacrés à ce sujet; ce sont:

en 1968: First International Conference of the Design Methods Group (32).

en 1971: Journées d'Informatique et Conception en Architecture organisées par l'IRIA.

en 1972: International Conference on Computers in Architecture (39).

en 1973: The Design Activity International Conference (37).

et les conférences de l'Environmental Design Research Association en 1969 (19), 1970 (21) et 1972 (31).

en 1974: CAD'74: International Conference and Exhibition on Computers in Engineering and Building Design (38).

Bibliographie

|1| ALEXANDER C.
Systems generating systems
AD / Dec. 68

|2| ARMOUR G. C., BUFFA E. S.
A heuristic algorithm and simulation approach to the relative location of facilities
Management Science / Vol. 9 / No. 2 / pp. 294-309 / 1963

|3| BERNHOLTZ A.
Quantification of the qualitative design process: Planning and Design by the computer
IFIP Congress 71 / Ljubljana / Aug. 71

|4| BERNHOLTZ A., FOSBURG S.
A generalized program for transforming relationship values into plan layouts
Proc. of Computer Graphics'70 / Brunel University / London / April 70

|5| BERNHOLTZ A., FOSBURG S.
Portrait of the computer as a young city planner
Interactive computer graphics / Brunel University / Sept. 72

|6| BIERSTONE E., BERNHOLTZ A.
RIDECS-RECOMP Procedure
Dept. of Civil Engineering / M.I.T. / March 67

|7| BIJL A.
Computer Aided Architectural Design
Procs. Computer Graphics 70 / Brunel University / London / April 70

|8| BILLON R.
La conception automatique dans les systemes modulaires
Journees informatique et conception en architecture / IRIA 71 / (et) GAMSAU / Vol. 1 / No. 4 / Mai 73

|9| BOUDIER J. P., CHARALAMBIDES S. et al
Analyse de programmes d'allocation spatiale
Centre de mathematiques, methodologie et informatique / Institut de l'Environment / Paris / Avril 73

|10| Computer Program descriptions for architects
Department of the Environment / Directorate of Research Requirements / September 1972

|11| CENTRE MMI
Analyse des donnees en architecture et en urbanisme
Institut de l'Environnement / 72

|12| CENTRE MMI
Notes methodologiques en architecture et en urbanisme
No. 2: Allocation Spatiale / Institut de l'Environnement / 73

|13| CORMACK R. M.
A review of Classification
Journal of the Royal Statistical Society / pp. 321-367 / 1971

|14| DAVID B.
Aspect graphique d'un systeme conversationel pour la conception architecturale assistee par ordinateur
Automatisme / no.4 / avril 1974

|15| DAVID B., REVIRO V.
An approach to computer aided architectural deign
CAD 74: International Conference and Exhibition on Computers in Engineering and Building Design
London / sept. 1974

|16| EASTMAN C.
Representation for Space Planning
CACM / Vol. 13 / No. 4 / pp. 242-250 / 1970

|17| EASTMAN C. M.
Heuristic algorithms for automated space planning
Procs 2nd International Conf. on Artificial Intelligence / Imperial College / London / Sept. 71

|18| EASTMAN C. M.
A preliminary report on a system for general space planning
CACM / Vol. 15 / No. 2 / Feb / 72

|19| EASTMAN C. M. / ed. SANOFF H. and COHN S. /
EDRA1 / Procs of the 1st annual Environmental Design Research Association Conference /
University of North Carolina / Chapel Hill / June 69

|20| EASTMAN C. M. / ed.MOORE G. T. /
On the analysis of the intuitive design processes
in: Emerging Methods in Environmental Design and Planning
M.I.T. Press / 1970

|21| EASTMAN C. M., ARCHEA / Ed. /
Proc. of EDRA 2 / Pittsbury / 1971

|22| ELRUS D. G.
A comprehensive computer-aided building design system
Rapport / Dep. of Civil Engineering / M.I.T. Press / 1972

|23| FRIEDMAN Y.
Pour une architecture scientifique
Pierre Belfond ed / 71

|24| INSTITUT DE L'ENVIRONNEMENT
Actes du colloque: Amenagement de l'environnement et traitement de l'information
Institut de l'Environnement / mars 72

|25| KREJCIRIK M.
Computer Aided Plant Layout
Computer Aided Design / Automn / 1969

|26| LEE K., CORREIRA R.
A mental health project programmed and designed with the aid of interactive computer graphics
Procs of the ACM-IEEE 9th Design Automation Workshop / June 72

|27| LEE K., STEWART E. D.
ARK2 - An implementable computer-aided design system
Proc. of the international conference on computers in Architecture / University of York /
September 1972

|28| MARCH L., STEADMAN Ph.
The geometry of environment (An introduction to spatial organisation in design)
RIBA Publications Ltd. / 1971

|29| MAROY J. P.
Une methodologie du projet en architecture: Le programme PROMET
I.R.I.A. / Departement d'informatique appliquee / Dec. 70

|30| MAROY J. P. PENEAU J. P.
Multivariate statistical analysis in architectural design
Proc. of the design activity international conference / August 73

|31| MITCHELL / Ed. /
Proc. of EDRA 3 / 1972

|32| MOORE G. T.
Emerging methods in Environmental Design and Planning
The M.I.T. Press / 1970

|33| NEGROPONTE N.
The architecture machine
M.I.T. Press / 70

|34| NEGROPONTE N.
Machine Recognition and inference making in computer aids to architecture
Rapport / M.I.T. / 1973

|35| NEGROPONTE N., GROISSER L.
The semantics of architecture machines
Architectural design / Vol. 9 / 7) / et Journees
Informatique et Conception en Architecture /
IRIA 71

|36| NEWMAN W. N.
An experimental program for architectural design
The computer Journal / May 66

|37| Proc. of the design activity international conference
August 1973

|38| Proc. of the International Conference and Exhibition on compu
in Engineering and Building Design
London / sept. / 1974

|39| Proc. of the International Conference on Computers in
Architecture
University of York / September 1972

|40| PFEFFERKORN C. E.
The design Problem Solver: a system designing equipment or
furniture layouts
Department of Computer Science / Carnegie-Mellon University /
Sept. 72

|41| PIETRI P.
Programme conversationnel d'aide a la conception
architecturale: CIMBLEZ
Journees Graphiques / IRIA 72

|42| REINSCHMIDT K. F.
Computer Methods for building system design
Proc. of ASCE National Meeting on Structural Engineering /
Baltimore / Maryland / Avril 1971

|43| RIVERO V.
Etude methodologique de l'aide a la conception dans les
projets architecturaux
Rapport du DEA / USMG-ENSIMAG / September 73

|44| SAHNOFF, COHN / Ed /
Proc. of EDRA 1 / Chapel Hill W.C. / 1969

|45| SCHNEIDER J. B.
Solving Urban location problems: Human intuition versus the
Computer
American Institute of Planners Journal / March 71

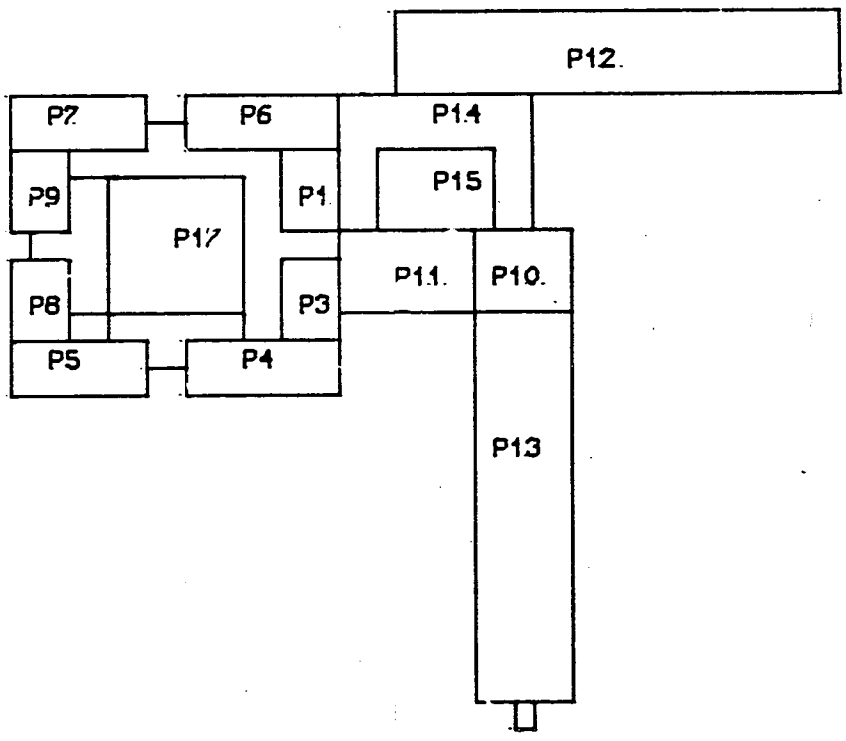
|46| WEINZAPSEL G.
The IMAGE System and its role in design
Proc. of the International Conference on Computers in
Architecture / University of York /
September 1972

|47| WILLIAMS J. H.
Computers Graphics in Architecture
Procs. of Computers Graphics 70 / Brunel University / London / April 70

|48| WILLOUGHBY T.
A rational approach to design
Architectural Association Quarterly / Oct. 70

|49| YESSIOS C. I.
FOSPLAN: a formal space planning language
EDRA-3 / ARC / Environmental Design Research Association Conference / Los Angeles / California / Jan. 72

|50| ZEITOUN U.
EXPOSE sur processus de conception et methodologie
Seminaire: Processus de conception et aide informatique en architecture / UPAG / Juin 1974



CONCLUSION

Nous pouvons résumer l'ensemble de ce qui a été exposé en précisant le sens exact du titre: étude d'un système pour la C.A.O. qui permette de concevoir des systèmes C.A.O. pour les applications variées; autrement dit un moyen de lutte contre l'inflation en C.A.O..

Nos travaux sur les schémas logiques et dans le domaine de la C.A.O. en architecture nous ont permis d'acquérir une certaine connaissance de la C.A.O.

Cette expérience nous a conduit à la conclusion que la C.A.O. est bien une discipline en soi et qu'elle a donc besoin d'outils propres aussi évolués que possible et dans la plupart des cas indépendants du domaine d'application particulier.

Nous avons donc, tout en continuant le travail assez technique sur les applications, commencé une étude plutôt méthodologique sur la C.A.O. dans un contexte interactif graphique, contexte qui nous paraît très important dans ce domaine.

Nous avons étudié d'abord les systèmes C.A.O. du point de vue des utilisateurs, puis nous avons dégagé une approche pour leur implémentation en distinguant les trois niveaux suivants:
structure du matériel, logiciel de base, outils évolués pour la C.A.O.

C'est le troisième niveau que nous nous sommes fixés comme objectif et pour cela nous avons étudié en détail les aspects importants d'un système C.A.O.: la formulation, l'integration de phases de conception et l'interaction homme-machine.

Nous nous sommes rendu compte qu'il ne suffit pas d'avoir les informations, mais qu'il faut les structurer et les encadrer pour pouvoir les traiter et les manipuler facilement dans les differents contextes.

Cela nous a conduit à dégager un formalisme (Multi-Sur-Réseaux), qui doit servir de structure d'accueil pour les informations et fournir les opérations associées.

Si on veut implémenter le système présenté il est indispensable de régler les problèmes concernant les deux premiers niveaux; le premier doit conduire à une configuration réaliste dans le contexte C.A.O.; le deuxième doit non seulement faciliter l'implémentation mais surtout la rendre efficace.

Le système proposé ne peut être vraiment justifié que si l'on doit s'occuper d'applications variées; leur conception et programmation constitueraient une dispersion d'efforts considérables que l'on se propose de limiter.

Tout au long de cette étude nous avons constamment vérifié la validité de notre démarche. C'est là que nous avons apprécié l'architecture, domaine riche en problèmes et en problèmes variés.

Actuellement les travaux sur SIGMA-Archi avancent très rapidement et nous n'avons volontairement pas présenté tous les aspects de ce système car les travaux notamment de RIVERO et Z'GRAGGEN doivent aboutir bientôt.

En ce qui concerne le système SIGMA-C.A.O. nous avons terminé un travail préliminaire de spécification et nous profitons de la présentation de cette thèse pour reccueillir le maximum d'avis sur l'approche choisie et le système proposé afin de savoir si le travail tel qu'il est décrit dans la deuxième partie du chapître II représente un certain intérêt et mérite, comme nous le pensons, d'être poursuivi.

Si tel est le cas, nous nous proposons de continuer sur deux plans:

-- les outils C.A.O.: il s'agirait d'implémenter le système SIGMA-C.A.O. avec, comme support matériel IRIS80 et peut-être le réseau CYCLADE, et comme système de base le système GERMINAL que nous devrions recevoir bientôt. Une fois notre système implémenté nous pourrions le tester et le raffiner.

-- les applications C.A.O.: notre expérience avec l'architecture s'est avérée très positive, c'est pourquoi nous pensons la

garder comme une application privilégiée. Il s'agirait de poursuivre les travaux sur le système SIGMA-Archi et avec une configuration du matériel pour un poste de travail autonome, nous pourrions établir un lien direct et effectif entre les deux systèmes: système SIGMA-C.A.O. générant le système SIGMA-Archi.