Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Comparaison de méthodes, de modèles et d'outils d'aide pour la conception de S.I.

Masson, Olivier

Award date:
1980

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 02. May. 2026

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

ANNEE ACADEMIQUE 1979-1980 .



Promoteur : F. BODART

Olivier MASSON

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

RUE GRANDGAGNAGE, 21, B - 5000 NAMUR (BELGIUM)

FACULTES
UNIVERSITAIRES
N.-D. DE LA PAIX
NAMUR

Bibliothèque
FMB16
1980/9

FM B16/1980/9

I 18/8/137

COMPARAISON DE METHODES, DE MODELES ET D'OUTILS D'AIDE POUR LA CONCEPTION DE S.I.

LBS3590699

77170

Avant l'exposé de ce mémoire, je tiens à remercier tous ceux qui, à titre divers, m'ont aidé à le réaliser.

Ma gratitude va tout d'abord à Monsieur le Professeur F.BODART pour avoir accepté la direction de ce travail. Je le remercie pour l'aide attentive qu'il m'a apportée tout au long de son élaboration.

Qu'il me soit permis également de remercier Monsieur Yves PIGNEUR, assistant de Monsieur BODART, pour les nombreux conseils qu'il m'a donnés.

Je prie Monsieur H. TARDIEU, du C.E.T.E. d'Aix en Provence, d'accepter ma plus vive reconnaissance pour m'avoir accueilli pendant 3 mois au sein de son équipe. Je tiens aussi à remercier Messieurs H. HECKENROTH et B.ESPINASSE, membres de son équipe,qui m'ont beaucoup aidé lors de l'étude de la méthode MERISE.

Enfin, que toutes les personnes que je ne peux citer et qui ont participé d'une quelconque façon à ce mémoire reçoivent ici l'expression de mes plus vifs remerciements.

o

o o

TABLE DES MATIERES

INTRODUCTION 1

1. Objet du mémoire 4

2. Contenu du mémoire 5

PREMIERE PARTIE : ANALYSE METHODOLOGIQUE DE MERISE ET 'NAMUR' 6

CHAPITRE I : ETAPES D'ANALYSE ET NIVEAUX DE DESCRIPTION 8

I 1. Les niveaux de description 8

I 11. Rappel 8

I 12. Commentaires 10

I 2. Les étapes d'analyse 12

I 21. Les grandes étapes d'un projet suivant MERISE 12

I 22. Etat des modèles lors des différentes étapes d'analyse MERISE 15

I 23. Les grandes étapes d'analyse : 'NAMUR' 15

I 24. Etat des modèles lors des différentes étapes d'analyse 'NAMUR' 16

CHAPITRE II : COMPARAISON DES MODELES : MERISE, 'NAMUR' ... ET AUTRES 17

II 1. Modèles de niveau conceptuel 17

II 11. MERISE : modèles conceptuels 17

. M.E.D. - M.C.D. 17

. Exemple de M.C.D. 22

. M.C.T. 24

. Exemple de M.C.T. 31

II 12. 'NAMUR' : modèles de niveau conceptuel 33

. S.C.D. 33

. Exemple de S.C.D. 38

. Modèle des traitements 40

. Exemple de modélisation des traitements 48

II 13. Commentaires 51

II 2. Modèles de niveau logique 55

II 21. MERISE : modèles logiques 55

. M.L.D. 55

. M.L.T. 57

Exemple de M.L.D. et de M.L.T. 64

II 22. 'NAMUR' : modèles de niveau logique 68

. Modèle des accès logiques 68

. Exemple de S.A.L. 74

. Modèle logique des traitements 77

. Exemple 81

II 23. Commentaires 83

CHAPITRE III : COMPARAISON DES CORRESPONDANCES INTER - MODELES 85

III 1. Rapports entre les modèles externes et conceptuels d'un S.I. 85

. Pour MERISE 86

. Pour 'NAMUR' 90

III 2. Mapping entre les niveaux conceptuel et logique de la description dù S.I. 92

. Pour MERISE 92

. Pour 'NAMUR' 96

III 3. Passage au niveau physique (p.m.) 98

III 4. Cohérence entre les modèles de données et les modèles de traitements 99

. Pour MERISE 99

. Pour 'NAMUR' 100

CHAPITRE IV : COMPARAISON DES METHODES PROPOSEES PAR MERISE ET 'NAMUR' 101

IV 1. Méthode MERISE : les étapes et leur articulation 102

IV 11. Elaboration des modèles de niveau conceptuel 102

IV 12. Elaboration des modèles de niveau logique 109

IV 13. Passage au niveau physique (p.m.) 121

IV 2. Méthode 'NAMUR' : les étapes et leur articulation 122

IV 21. Description conceptuelle du S.I. 122

IV 22. Description de niveau logique du S.I. 126

IV 23. Passage au niveau physique de la construction du S.I. (p.m.) 132

IV 3. Commentaires 132

CHAPITRE V : OUTILS DEVELOPPES DANS LE CADRE DES DEUX METHODES 135

V 1. Outils proposés par MERISE 135

V 2. Outils proposés par 'NAMUR' 149

DEUXIEME PARTIE : DEVELOPPEMENT PARTICULIER 155

CHAPITRE VI : COMMENT D.S.L. POURRAIT EXPRIMER LES CONCEPTS DES MODELES DE MERISE 156

VI 1. Comment D.S.L. pourrait exprimer les concepts du M.C.D. MERISE ? 157

VI 2. Comment D.S.L. pourrait exprimer les concepts du M.C.T. MERISE ? 161

VI 3. Comment D.S.L. pourrait exprimer les concepts du M.L.T. MERISE ? 172

CONCLUSION 180

BIBLIOGRAPHIE

0. Introduction.

Depuis plusieurs années, le 'management' des organisateurs se complexifie de façon telle qu'il devient de plus en plus difficile d'assurer un fonctionnement cohérent de l'ensemble de l'organisation tout en s'efforçant de respecter les objectifs fondamenteaux qu'elle s'est assignée, et de maîtriser les nombreuses interactions dont elle est le siège. Parmi les causes de cette complexification, citons :

- la taille de plus en plus importante des organisations tant publiques que privées,
- le cloisonnement artificiel au sein de celles-ci,
- l'élargissement du domaine de relations des organisations grâce aux différentes conventions internationales (passage du marché national au marché européen voire mondial),
- l'évolution constante de l'environnement des organisations,
- le manque de plus en plus flagrant de motivation de la part du personnel face à une spécialisation souvent outrancière des tâches et le peu d'initiative qu'il lui est accordé,
- l'instabilité qui caractérise la conjoncture actuelle.

Face à ces difficultés, les gestionnaires se trouvent confrontés à de gros problèmes dans leur tâche d'analyse et de décision. De plus, à un 'pilotage quasi automatique' de l'organisation qui caractérisait les périodes de forte stabilité et basé presqu'exclusivement sur des informations découlant d'événements passés succède maintenant 'un pilotage à vue' réclamant des informations plus élaborées et plus précises, informations de synthèse et de prévision qu'on doit être en mesure de fournir aux décideurs.

Les systèmes d'information (S.I.) et plus particulièrement les systèmes d'information automatisés (1) (S.I.A.) sont des outils qui peuvent s'avérer utiles à la gestion d'une organisation.

Voyons donc quelle est leur place et leur rôle dans l'organisation.

---

(1) Par la suite, quand nous parlerons du S.I., il s'agira en fait de S.I.A., les seuls qui nous intéressent dans le cadre de ce mémoire.

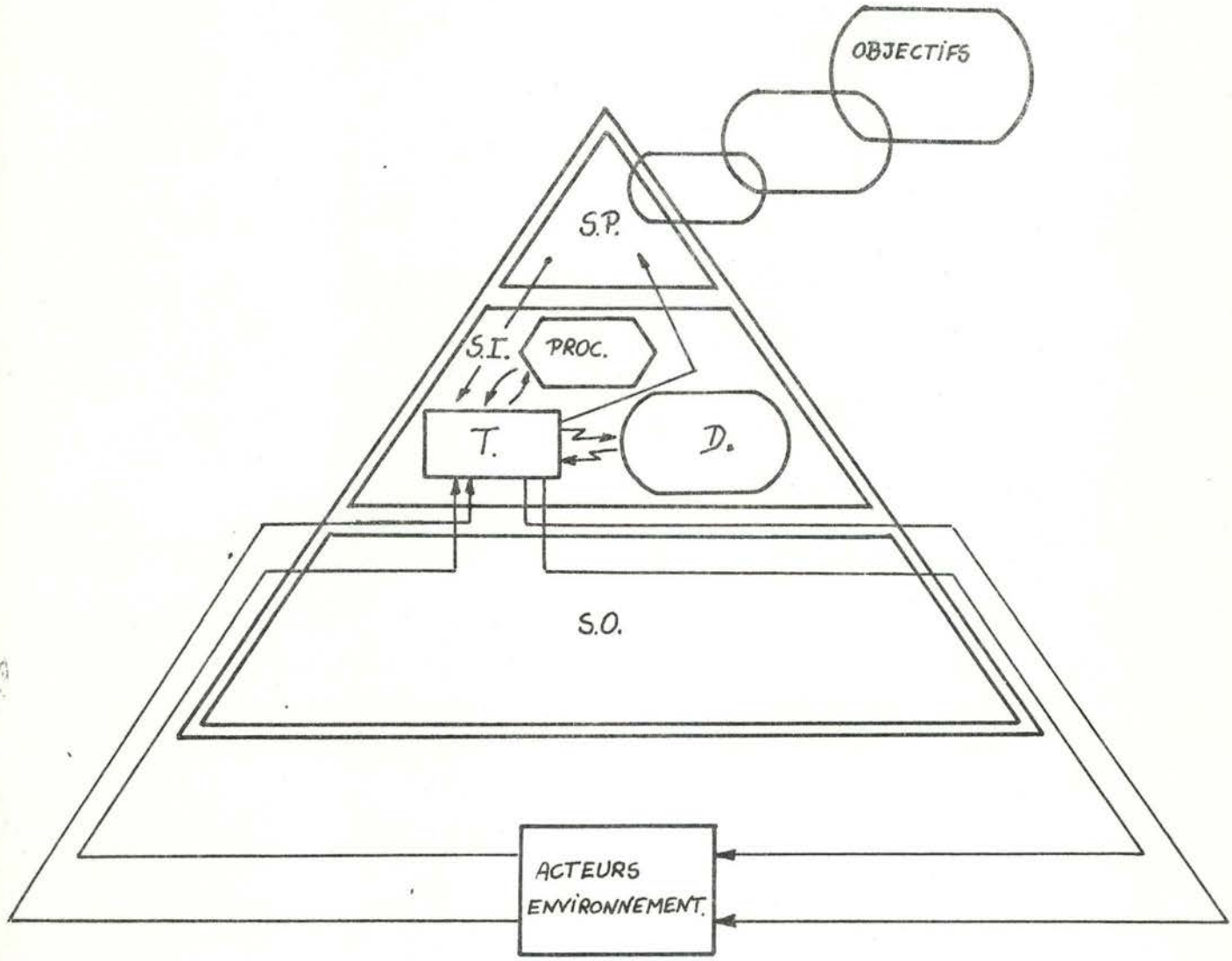


fig 0.

Comme l'ont montré K.E. BOULDING et J.L. Le MOIGNE, une organisation sociale, qui peut être considérée comme un système, est composée de 3 (sous-) systèmes :

a) le système opérant S.O. : agit, fonctionne et processe les flux physiques qui traversent l'organisation (flux logistique : biens et services, flux de personnel, flux monétaire, flux d'actifs)

b) le système de pilotage S.P. : dirige, coordonne l'action du

S.O. en fonction - des objectifs fondamentaux de l'organisation,
- de l'évolution de l'environnement,
- de la connaissance qu'il a du comportement du S.O.

c) le système d'information S.I. : permet le couplage entre le S.P. et le S.O.
Il comprend un ensemble synchronisé de traitements : T qui consulte et met à jour un ensemble de données mémorisées : D à l'aide d'un ensemble de processeurs (hommes et/ou machines)
Le S.I. est en communication :

• avec le S.O. :
- il garde en mémoire une représentation de l'activité du S.O. au sein de L'organisation,
- il transmet au S.O. les décisions du S.P.,

• avec le S.P. :
- il transmet au S.P. les informations qu'il a éventuellement traitées et qui sont nécessaires à la prise de décision,
- il reçoit du S.P. les décisions à transmettre au S.O. ou à l'environnement de l'organisation,

• avec l'environnement de l'organisation.

Rmq. : Le système d'information automatisé (S.I.A.) comporte les mêmes éléments que le S.I. dans lequel il est contenu. Il peut coïncider avec le S.I. (système entièrement automatisé) ou se réduire à rien (système entièrement manuel). Dans ce travail, nous nous sommes intéressés uniquement au S.I.A. qui agit sur des informations formalisables et codifiables.

Jusqu'à présent, les S.I. étaient limités
- soit à un nombre très élevé de fichiers sur lesquels travaillaient quelques programmes assez élémentaires,
- soit à des chaînes dites intégrées engendrant une multitude de fichiers de liaison et nécessitant une exploitation des

programmes suivant un ordre et un calendrier précis inadaptée à la période d'instabilité que l'on connaît actuellement.

Les méthodes de conception de S.I. doivent désormais prendre en considération des facteurs aussi bien techniques que socio-économiques :

facteurs techniques :

- les bases données et les S.G.B.D. qui facilitent, entre autres, l'accès direct à des données suivant des critères multiples et permettent de gérer des biens entre des ensembles de données,
- le développement des réseaux et de l'informatique répartie,
- la mini et la micro-informatique,
- la généralisation de l'emploi de terminaux et de systèmes interactifs,

facteurs socio-économiques :

- la décentralisation des décisions,
- l'enrichissement des tâches,
- l'organisation rationnelle des postes de travail,
- la sécurité et la confidentialité,
- l'accroissement des possibilités de 'pilotage' et d'amélioration des processus de gestion,

et elles doivent aussi permettre :

- d'associer étroitement les aspects organisationnels et informatiques,
- d'accroître la qualité des relations entre utilisateurs et informaticiens,
- de facilité le dialogue à tout moment (étude, conception, formalisation, réalisation...).

01. Objet du mémoire.

L'objet de ce mémoire sera principalement de comparer deux méthodes de conception de S.I., les modèles qu'elles préconisent et les outils qu'elles développent.

Une de ces méthodes est étudiée à Aix en Provence par H. Tardieu et H. Heckenroth : MERISE, l'autre est celle qui est proposée aux étudiants des F.N.D.P. de Namur et que j'appellerai durant tout ce travail, pour des raisons de facilité : 'NAMUR'.

MERISE :

MERISE entend . proposer une approche globale aux S.I.,
. distinguer les domaines 'données' et 'traitements',
. prendre en compte les paramètres techniques et socio-économiques,
. et fournir un outil méthodologique souple permettant une certaine adaptabilité suivant les particularités des futurs utilisateurs de la méthode.

Elle s'attache à donner un support continu et adapté à la conduite d'un projet prenant en compte les évolutions prévisibles de l'informatique.

'NAMUR' :

'NAMUR' est ce que nous pourrions appeler 'une méthode virtuelle' puisqu'elle est plus le résultat de la fusion des idées d'un certain nombre de personnes à l'Institut d'Informatique de Nâmur sur la conception et la réalisation des S.I. (1) qu'une méthode développée de bout en bout par une équipe de chercheurs. 'NAMUR' poursuit les mêmes objectifs que MERISE, à savoir fournir aux utilisateurs une méthode, des modèles et des outils d'aide pour la conception et la réalisation de S.I. tout en tenant compte des différents facteurs (techniques, socio-économiques et autres) énoncés précédemment.

02. Contenu du mémoire.

Ce travail comprendra deux parties :

- la première, la plus importante, sera une analyse méthodologique de MERISE et de 'NAMUR' (2);
- la seconde, quant à elle, sera un développement particulier de la première partie et montrera comment un langage développé dans l'approche 'NAMUR' est capable d'exprimer les concepts de MERISE.

---

(1) C' est ainsi que nous reprendrons, dans la suite, les résultats de travaux effectués notamment par Mrs F. BODART et J.L. HAINAUT.

(2) MERISE et 'NAMUR' sont toujours en cours d'élaboration. Cependant, à travers des données fragmentaires, nous essayerons de dégager une unité d'ensemble de ces deux démarches.

P R E M I E R E P A R T I E

ANALYSE METHODOLOGIQUE DE MERISE ET DE 'NAMUR'.

Cette première partie se décompose en cinq chapitres :

Dans un premier chapitre, nous rappelons quels sont les différents niveaux de description et d'analyse d'un S.I.

Le deuxième chapitre décrit les modèles utilisés pour la description conceptuelle et logique d'un S.I. et ce, pour MERISE et 'NAMUR'.

Au cours du troisième chapitre, nous étudions les différents procédés de transformation et de 'mapping' entre les niveaux de description conceptuel et logique.

Le quatrième chapitre s'attache à détailler les méthodes MERISE et 'NAMUR' proprement dites.

Enfin, au chapitre cinq, nous considérons les outils d'aide à la conception de S.I. développés par les deux approches.

Chap. I. Etapes d'analyse et niveaux de description.

I 1. Les niveaux de description.

I 11. Rappel.

Le groupe ANSI - SPARC est le premier à avoir explicitement montré la nécessité de 3 niveaux différents de perception dans la modélisation d'une base de données.

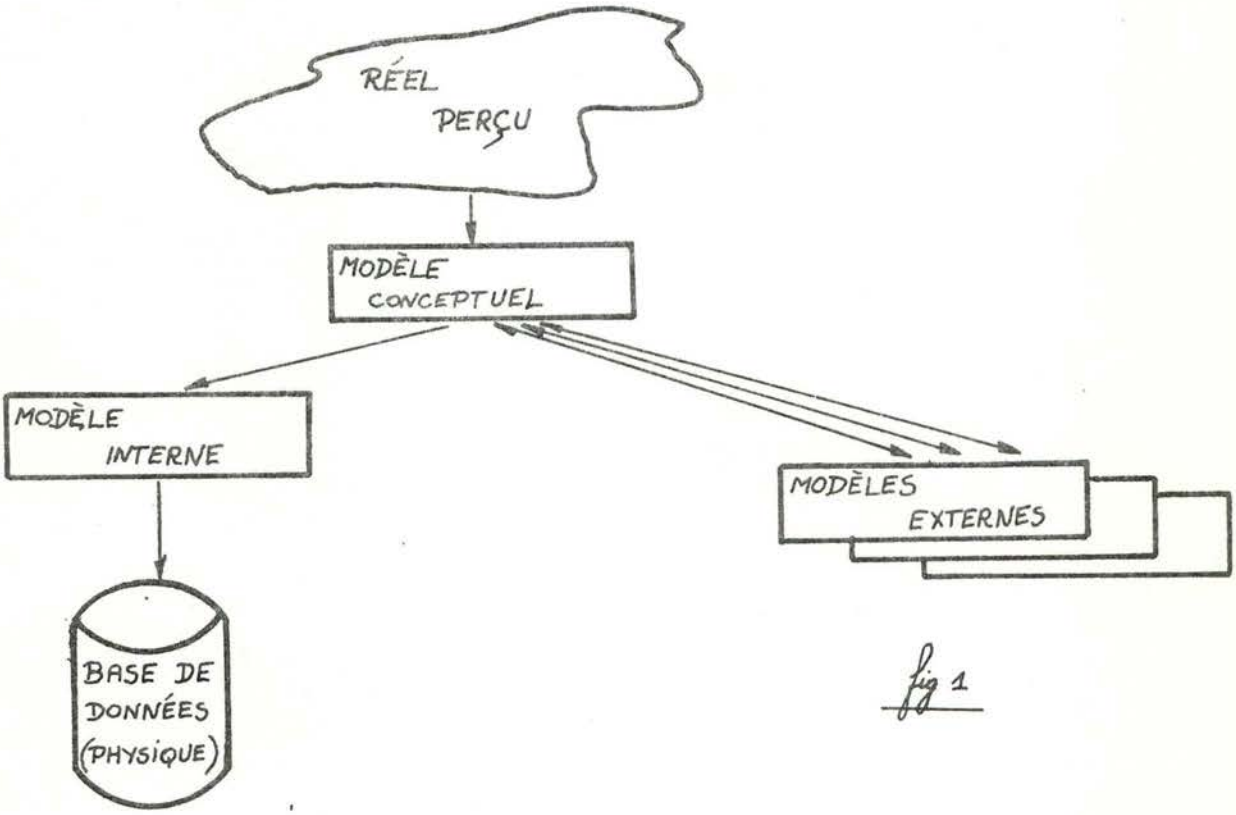


fig 1

ANSI - SPARC a souligné la difficulté qu'il y avait à décrire directement le réel perçu avec le formalisme informatique c.-à-d. traduire la réalité dans le langage de description de données (L.D.D.) des S.G.B.D.
Ils ont donc introduit plusieurs niveaux dans le processus de structuration d'une B.D. :

le réel perçu : représentation que le système de décision se construit pour son usage de l'activité de l'organisation au sein de son environnement (activité définie par le système de décision en fonction des grands objectifs fixés).

Cet ensemble d'informations, éventuellement reliées entre elles, est aussi le résultat de la perception des différents acteurs de l'organisation.
Il est exprimé, à ce niveau, dans un langage courant de l'organisation : fiches, bordereaux, ...

le niveau conceptuel : niveau charnière entre le réel perçu et le niveau interne, il représente le passage de la diversité des acteurs et des utilisations à la stabilité et à l'unicité des informations. On distingue :

le modèle conceptuel : unique pour l'organisation, il est la traduction du réel perçu. Il contient la description, par l'intermédiaire des propriétés, des objets stables pour l'organisation et des relations entre ces objets, ainsi que des informations à mémoriser.
Il est exprimé dans un langage rigoureux mais suffisamment naturel pour être compris et utilisés par des non-informaticiens.

le modèle externe : pouvant être exprimé dans un langage différent de celui utilisé pour le modèle conceptuel (1), il représente une utilisation particulière de l'information mémorisée.
La structure des informations du (ou des) modèle(s) externe(s) peut être différente de celle du modèle conceptuel mais doit être compatible avec cette dernière; une étape

---

(1) Il existe donc un processus de passage des modèles externes au modèle conceptuel : MAPPING.
Il faut signaler que dans les approches MERISE et 'NAMUR', les modèles conceptuel et externes sont exprimés dans le même langage, ce qui a pour effet de simplifier énormément le mapping entre ces modèles.

de validation sera donc parfois nécessaire afin de vérifier cette compatibilité.

le niveau interne : il n'existe, à un moment donné, qu'un seul modèle interne. Il définit la réalisation de la structure de données exprimée par le modèle conceptuel.

Il existe donc, pour ANSI - SPARC, 3 étapes dans la conception d'une B.D. :

1) élaboration du modèle conceptuel à partir du réel perçu,
2) élaboration des différents modèles externes et mappings de ces derniers avec le modèle conceptuel,
3) passage du modèle conceptuel au modèle interne.

I 12. Commentaires.

Si le rapport ANSI - SPARC est très explicite en ce qui concerne la partie 'données' d'un S.I. (conception et réalisation de la B.D.), il passe complètement sous silence la partie relative aux traitements, la représentation et la structuration des différents facteurs du traitement, ainsi que l'articulation de ces traitements avec les structures de données choisies. Les approches MERISE et 'NAMUR' reprendront cette découpe en niveaux - explicitant néanmoins le niveau interne en le découpant en niveaux logique et physique - et l'étendront au formalisme des traitements.

On aura ainsi 3 niveaux de description :

- le niveau conceptuel,
- le niveau logique,
- le niveau physique,

et ce, aussi bien pour les données que pour les traitements.

Le niveau conceptuel : il sera la prise en compte du monde réel tel qu'il est perçu par l'organisation, des objectifs et des choix de gestion de l'organisation. Alors que la description de niveau conceptuel des données cherchera à donner un sens à chacun des éléments d'une vision statique de l'univers dans lequel l'organisation exerce son activité, la

description de niveau conceptuel des traitements s'attachera à donner une vision dynamique de l'objet de cette activité.
Il sera exprimé sous une forme qui se veut manipulable et compréhensible pour les différents responsables de l'organisation, même non-informaticiens.

le niveau logique : traduction fidèle du niveau conceptuel, il sera la charnière entre ce dernier et le niveau physique.
Il précisera les traitements décrits au niveau conceptuel et prendra en compte les choix organisationnels faits par les responsables en donnant l'affectation de ces traitements aux différents types de processeurs.
Quant à la partie 'données', elle sera transposée afin de mettre en évidence les hiérarchies d'accès logiques qu'il sera possible ensuite d'optimiser.

le niveau physique : il découlera du niveau logique et prendra en charge les choix techniques pour fournir un système opérationnel.
A ce niveau donc, nous aurons la B.D. complète ainsi que les différents programmes qui l'utiliseront.

A chacun des niveaux sont attachés des concepts spécifiques et précis regroupés en modèles (1).
Ces modèles constituent le guide permettant de décrire tous les éléments associés à un niveau donné sachant que des règles lient chacun des concepts dans un niveau mais aussi d'un niveau à l'autre.

---

(1) Le terme "modèle" sera dans la suite pris :
soit dans le sens : ensemble des concepts propres à un formalisme et à un niveau,
soit dans le sens : description même de l'organisation en utilisant les concepts propres à un formalisme et à un niveau.

I 2. Les étapes d'analyse.

Il faut également signaler qu'à chaque niveau de description, les concepts sont regroupés en 2 domaines d'intérêt distincts :

- celui des données,
- celui des traitements.

Ainsi, en résumé, nous avons :

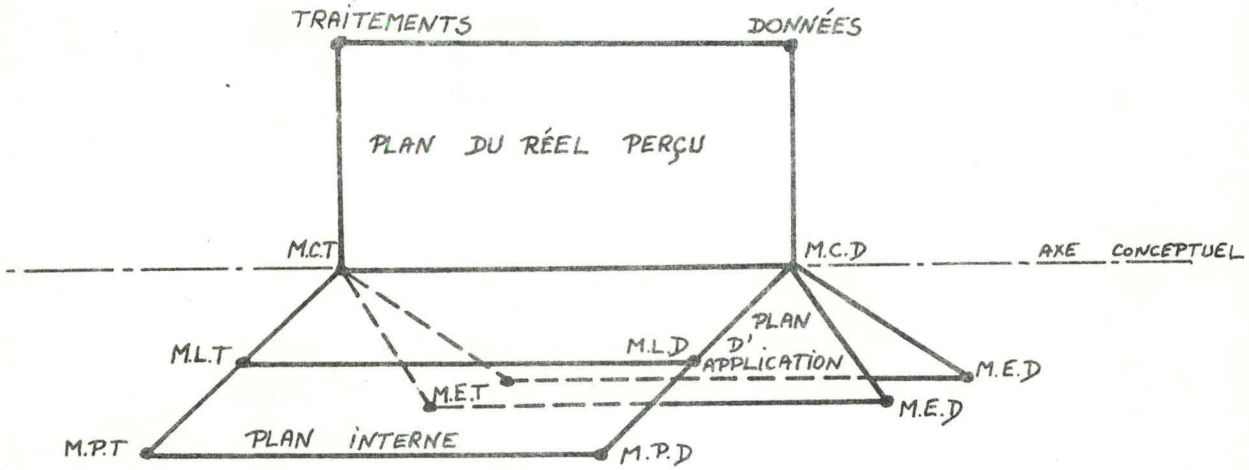


fig 2

I 2. Les étapes d'analyse.

Des méthodes d'analyse très différentes peuvent utiliser une architecture de modèles identique (en l'occurrence ici, les modèles conceptuel - logique - physique).
Parmi ces différentes démarches d'analyse, nous n'en étudierons que deux : celles proposées par MERISE et 'NAMUR'. (1)

I 21. Les grandes étapes d'un projet suivant MERISE.

Elles sont en fait déterminées par les niveaux de décisions qui interviennent tout au long du projet (création ou modifi-

---

(1) Il en existe d'autres : cfr.

- Tom de Marco : "Structured Analysis and System Specification".
- Victor Weinberg : "Structured Analysis".
- G.M. Nijssen : "A Gross Architecture For The Next Generation Database Management Systems".

cation d'un S.I.) et non par le découpage en niveaux d'intérêt conceptuel - logique - physique.

On distingue :

- l'étude préalable : lancée en fonction d'un besoin pressenti ou conformément à un schéma directeur, elle étudie les solutions fonctionnelles et techniques par rapport à un existant et des orientations, et elle évalue les coûts du nouveau système en exploitation, les coûts et les délais de réalisation.
  En fin d'étude préalable,
  - soit on rejettera le projet,
  - soit on modifiera le champ de l'étude et on retournera alors à l'étude préalable,
  - soit on continuera l'étude en définissant les solutions fonctionnelles et techniques et en découpant éventuellement en sous-projets.

- l'étude détaillée : il s'agit ici - d'obtenir une décision de chacun des utilisateurs sur les spécifications externes détaillées relatives au sous-projet qui le concerne,
  - d'affiner les évaluations des coûts et confirmer les choix en ce qui concerne le matériel et l'exploitation.

- la réalisation : termine la description d'un sous-projet et la met sous une forme interprétable par une machine.
  C'est à la fin de cette étape que l'on prendra la décision de la mise en oeuvre du sous-projet (formation des utilisateurs, constitution des fichiers ou de la B.D., mise en place des matériels,..)
  Dès que le sous-projet a été mis en oeuvre et que des essais en réel ont pu être effectués, on peut prendre la décision finale en fonction de la "recette" définitive (gain enregistré par la mise en service du sous-projet) : le sous-projet sera alors retenu ou rejeté.

Ces différentes étapes peuvent être résumées comme suit :

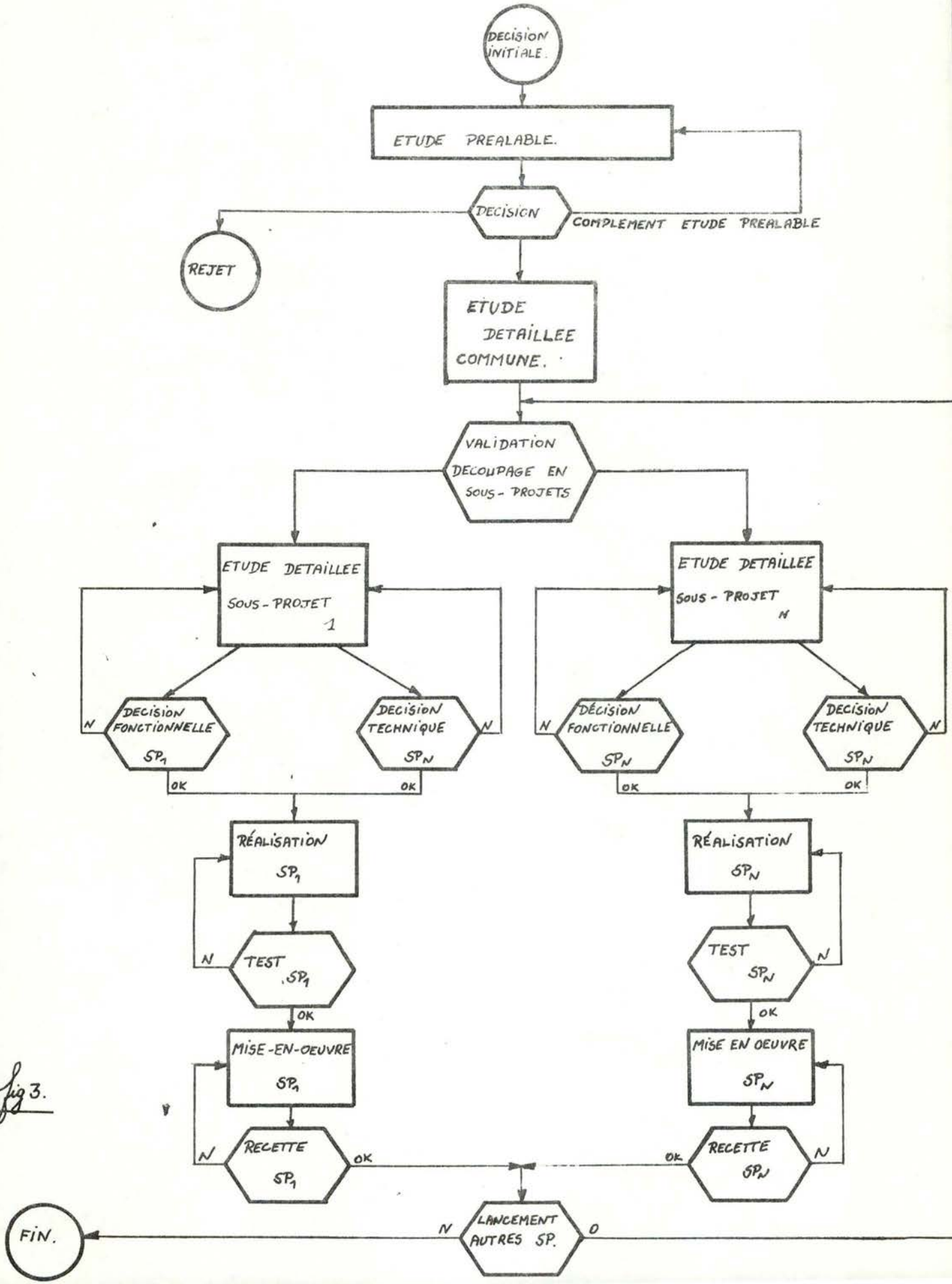


fig 3.

I 22. Etat des modèles lors des différentes étapes d'analyse (MERISE)

- L'étude préalable ébauche le M.C.S., le M.L.S. et le M.P.S. (1) pour élaborer un dossier de choix.
- L'étude détaillée affine le M.C.S. et fournit un M.L.S. presque complet.
- La réalisation précise le M.L.S. et donne un M.P.S.
- La mise en oeuvre optimise le M.P.S.

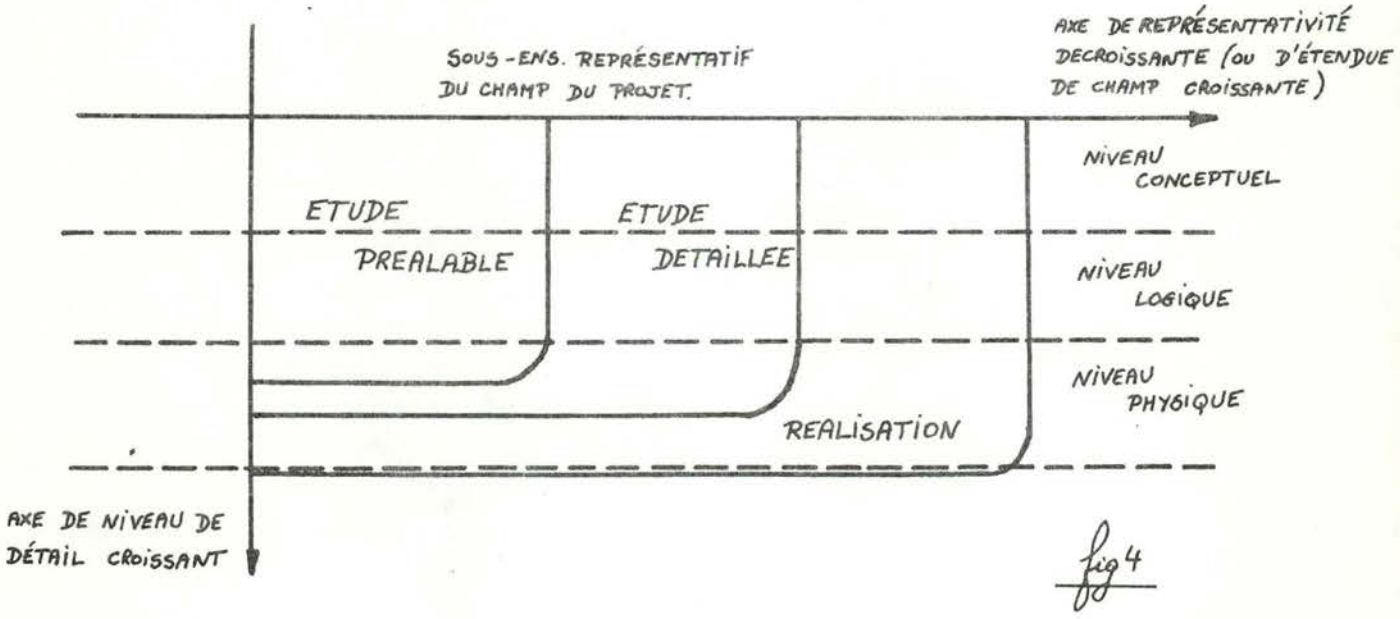


fig 4

I 23. Les grandes étapes d'analyse (2) : 'NAMUR'

L'analyse et la description d'un S.I. est précédée d'une étude d'opportunité. Cette étude est entreprise dans le cadre d'un plan global de développement de l'informatique au sein de l'organisation ("plan informatique","schéma directeur", ...). Elle définit l'objet de la future analyse, comporte une part essentielle d'évaluation des coûts et bénéfices et se termine par la rédaction d'un cahier des charges.
On distingue alors deux étapes dans l'analyse détaillée de la solution retenue par l'étude d'opportunité :

- l'analyse fonctionnelle : s'attache à décrire de façon

---

(1) M.C.S., M.L.S., M.P.S. } modèles { conceptuel, logique, physique } du système (données + traitements)

(2) A. Clarinval : "Méthodologie de l'analyse".

rigoureuse et détaillée les structures de données et de traitements ainsi qu'à fournir toute une série de quantifications. Elle est particulièrement attentive aux besoins et directives des utilisateurs.

- l'analyse organique : a pour but l'implémentation, en vue de l'exploitation, des solutions logiques dégagées au cours de l'analyse fonctionnelle. Elle détermine principalement la forme concrète de la B.D. et des programmes en mettant en oeuvre un ensemble de démarches informatiques techniques.

I 24. Etat des modèles lors des différentes étapes d'analyse ('NAMUR')

- L'étude d'opportunité se base sur des éléments appartenant aux 3 niveaux.
- L'analyse fonctionnelle fournit le M.C.S. ainsi qu'une partie du M.L.S.(notamment toutes les considérations sur la répartition des traitements aux différents processeurs, mais pas encore les structures d'accès aux données)
- L'analyse organique complète le M.L.S. (principalement le schéma des accès logiques) et élabore le M.P.S.

Ainsi donc, nous avons

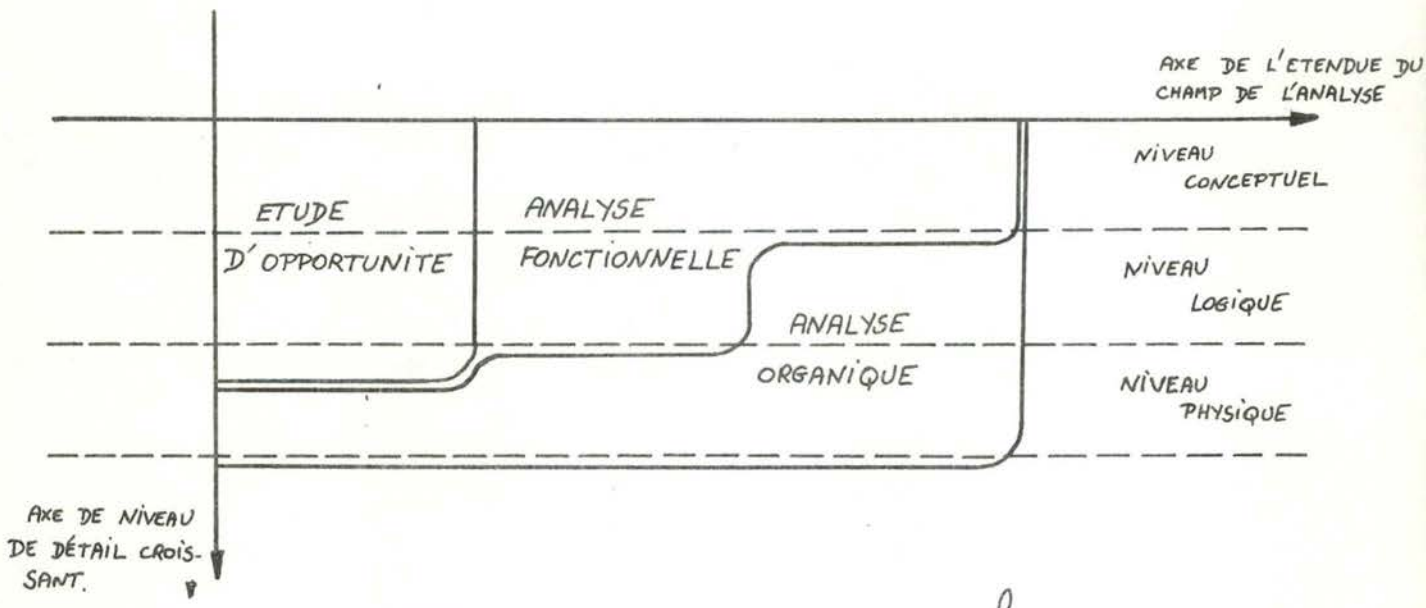


fig 5

Chap. II Comparaison des modèles : MERISE, 'NAMUR'... et autres.

Nous allons comparer, dans ce chapitre, des modèles c'est-à-dire des ensembles de concepts qui doivent permettre de décrire l'organisation et son activité. Cette description se fera en plusieurs niveaux, comme nous l'avons vu au chapitre précédent, et à chaque niveau, elle utilisera deux modèles : - celui des données,
- celui des traitements;
couvrant ainsi et le côté statique et le côté dynamique de l'organisation.

Délaissant volontairement le niveau physique trop tributaire des choix techniques, nous nous attarderons principalement sur les modèles conceptuels et logiques proposés par les deux méthodes considérées et éventuellement sur d'autres modèles préconisés par ailleurs.

II 1. Modèles de niveau conceptuel.

II 1.1. MERISE : modèles conceptuels.(3)

• M.E.D. - M.C.D. : modèles de données:

Le modèle conceptuel de données M.C.D., ainsi que tous les modèles externes de données M.E.D. - sous-ensembles sémantiques du M.C.D- se base sur le même jeu de concepts et de règles. (fig. 6.)

3 éléments descriptifs fondamentaux :

Propriété:

- Particule élémentaire d'information, elle permet de décrire les individus-types (1) et les relations-types (2)(1).

---

(1) cfr. ci-après.

(2) chaque niveau de description est réalisé en termes de types; les occurences ne sont prises en charge qu'au stade d'exécution.

(3) Les définitions des concepts des différents modèles de MERISE s'inspirent du 'Glossaire pour les données et les traitements' H. TARDIEU et H. HECKENROTH.

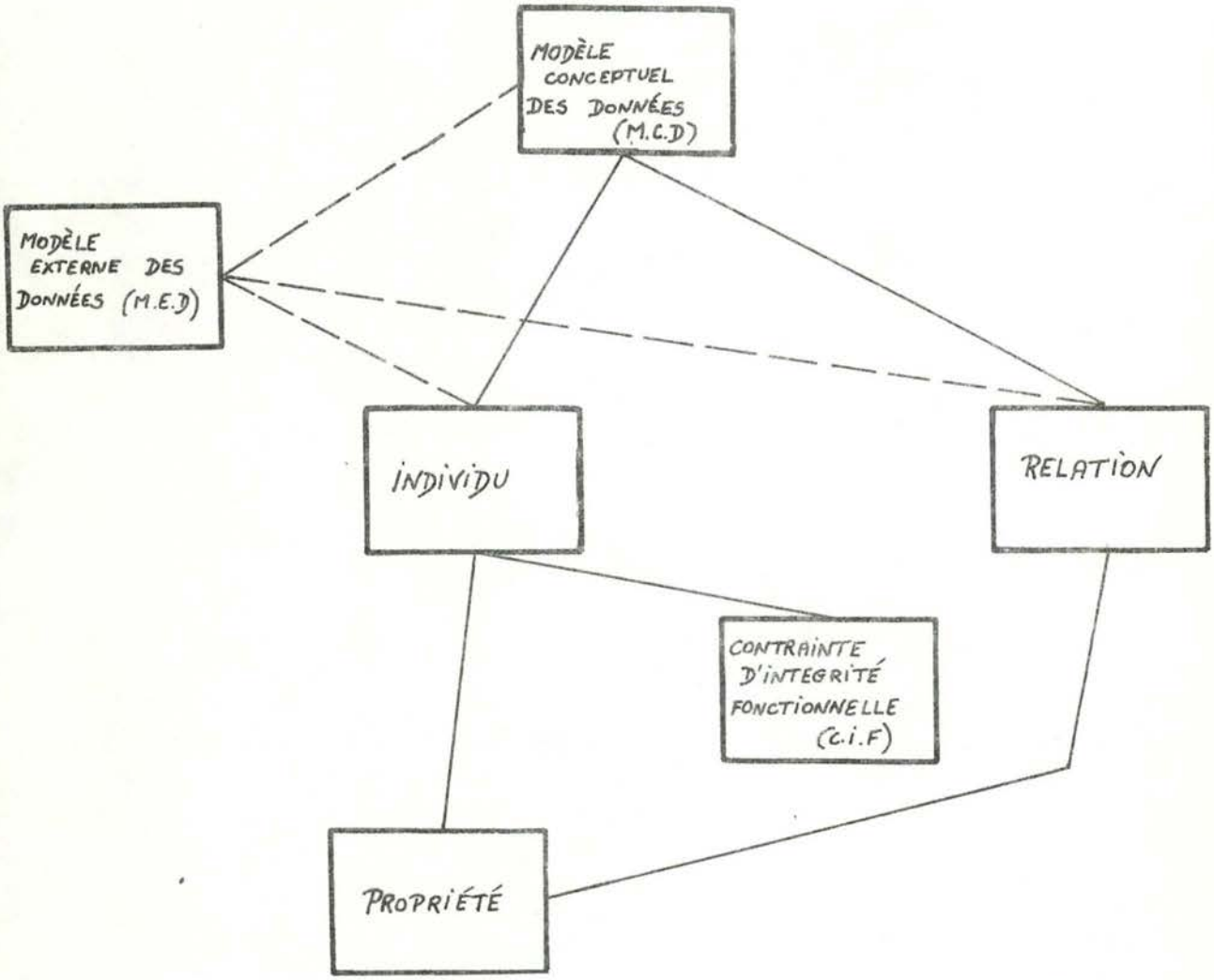


fig 6

- Les occurences de propriété sont des valeurs d'un domaine fini (ex. : sexe : M. ou F.) ou non (ex. : prix) .
- Chaque type de propriété ne peut appartenir qu'à un individu-type (1) ou à une relation-type (1) même si un domaine de valeurs peut être commun à plusieurs propriétés.
  On peut éventuellement distinguer :
  - une propriété élémentaire : plus petit élément logique d'information manipulé dans un système;
  - une propriété composée : regroupement logique de propriétés élémentaires formant une nouvelle information sémantiquement différente de ses constituants.

---

(1) cfr. ci-après.

Individu :

- Un individu-type est une famille d'objets concrets ou abstraits que l'on a choisi de tous caractériser par la même liste de propriétés sur toute l'étendue du réel perçu.

- Son choix est motivé par l'intérêt qu'on lui porte dans l'organisation. Il doit être défini sans faire référence à d'autres objets et chacune de ses occurences doit être distinguable des autres, doit être dotée des mêmes propriétés sur toute l'étendue du réel perçu (toutes les propriétés doivent avoir un sens pour toutes les occurences de l'individu-type) et, pour chaque propriété, ne doit admettre qu'une seule valeur.

- Tout individu-type doit pouvoir être identifié par une au moins de ses propriétés (identifiant) dont les valeurs permettront de distinguer sans ambiguité toutes les occurences de cet individu.

  Rmq. : on appellera entité, une liste de propriétés d'un individu-type (en général, on n'en définira qu'une, celle qui reprendra l'ensemble des propriétés de l'individu-type) .

Relation :

- Une relation est une classe d'informations définie sur un ensemble d'individus-types et stable sur l'étendue du réel perçu. Elle comporte un certain nombre de propriétés qui n'ont de sens que par rapport à l'association des individus-types; elle n'a donc pas d'existence propre et ne peut être insérée dans un modèle que si les individus-types qu'elle associe existent déjà.

- Chaque occurence doit être distinguable à l'intérieur d'une même relation-type, doit être dotée d'une même liste de propriétés (liste qui peut être vide, ce qui aurait pour signification que la relation n'exprime qu'un lien sémantique entre individus) , et ne peut admettre, pour chaque propriété, qu'une et une seule valeur à une date donnée.

- La relation-type a pour identifiant la concaténation des identifiants des individus-types qu'elle associe.

Rmq. : on appellera collection, la liste des individus-types qui composent la relation-type, leur nombre sera appelé dimension.
L'entité d'une relation-type sera quant à elle constituée par la réunion de la liste de ses identifiants et de la liste de ses propriétés propres.

Cardinalités et contraintes d'intégrité :

Aux trois concepts fondamentaux que nous venons de décrire, s'ajoutent un ensemble de contraintes :

- contraintes sur les propriétés :

  -d'ordre syntaxique : elles portent sur les formats d'écriture des propriétés;

  -d'ordre sémantique : elles portent sur l'ensemble des valeurs possibles pour une propriété.

- contraintes sur les relations :

  -cardinalité individuelle d'un individu-type au sein d'une relation-type : nombre de fois minimum et maximum qu'une même occurence de cet individu-type peut apparaître dans des occurences de la relation-type.
  Chaque fois qu'un individu-type participe à la collection d'une relation-type, il faut définir sa cardinalité individuelle.

  -cardinalité ou contrainte d'intégrité fonctionnelle (C.I.F.) : elle signale l'existence d'une dépendance fonctionnelle entre une sous-collection et un individu-type d'une même relation-type.
  Cela veut dire qu'à toute occurence de la sous-collection ne peut correspondre qu'une seule occurrence de l'individu-cible .

Rmq. : dans une relation binaire, les cardinalités individuelles 0-1, 1-1 entraînent l'existence d'une contrainte d'intégrité fonctionnelle.

Représentation graphique des concepts du M.C.D.:

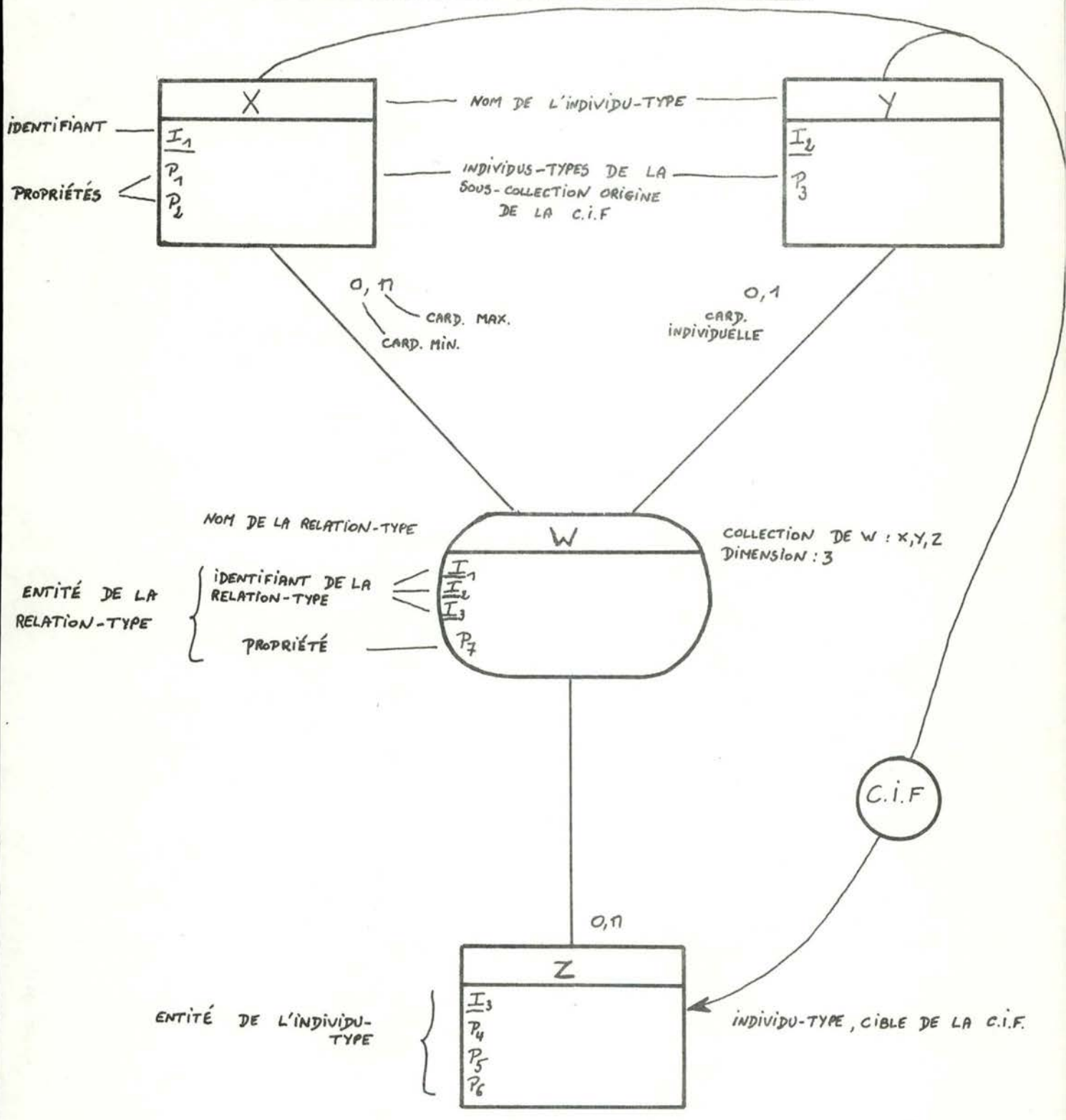


fig 7

Exemple de M.C.D. MERISE

Nous allons illustrer les concepts des différents modèles de MERISE (et de 'NAMUR') par un exemple. Nous reprendrons, à cet effet, le cas étudié à l'Institut d'Informatique de Namur par l'équipe de Monsieur BODART : le cas "PETITPAS". Il s'agit d'automatiser le traitement des commandes des clients, des livraisons aux clients et des réapprovisionnements dans la firme PETITPAS S.P.R.L. (1), firme de vente par correspondance d'articles d'habillement.
En voici donc le M.C.D. MERISE (construit à partir d'une version simplifiée du cas, retenue par J.L.HAINAUT dans "Analyse du cas "PETITPAS" - Dossier d'analyse organique - Première partie : la base de données). Cet exemple n'aura ici pas d'autre prétention que d'illustrer les concepts des modèles vus.

---

(1) Pour un énoncé complet du cas "PETITPAS" : cfr. :
- Rapport d'entretien,
- Rapport d'entretien (suite) : les quantifications,
- Proposition d'automatisation,

Institut d'Informatique, Namur.

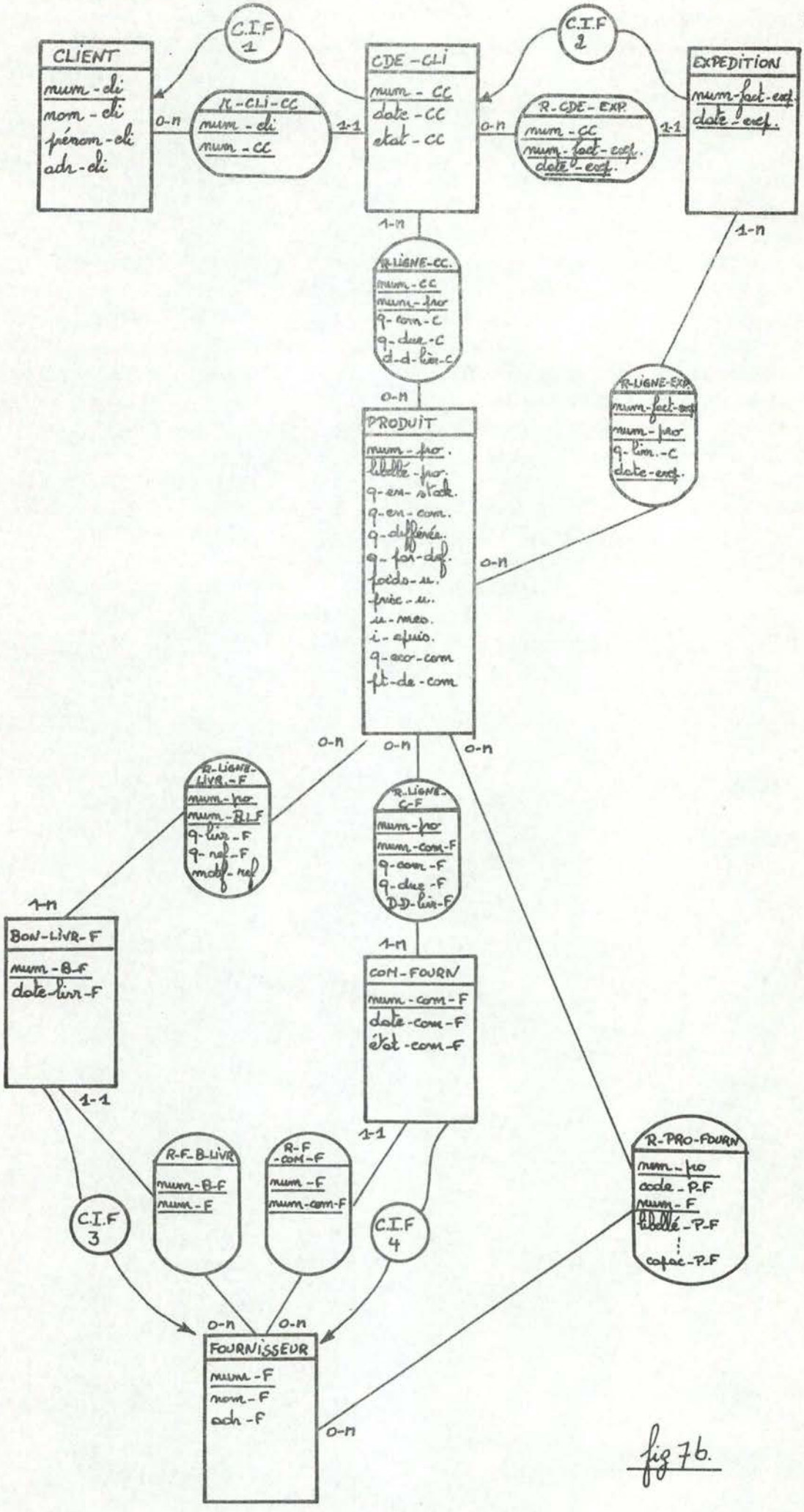

CLIENT
num-cli
nom-cli
prénom-cli
adr-cli
C.I.F 1
R-CLI-CC
num-cli
num-CC
0-n
1-1
CDE-CLI
num-CC
date-CC
etat-CC
C.I.F 2
R-CDE-EXP
num-CC
num-fact-exp.
date-exp.
EXPEDITION
num-fact-exp.
date-exp.
1-n
0-n
1-1
1-n
R-LIGNE-CC.
num-CC
num-pro
q-com-C
q-due-C
d-d-liv-C
0-n
R-LIGNE-EXP.
num-fact-exp
num-pro
q-liv.-C
date-exp.
PRODUIT
num-pro.
libellé-pro.
q-en-stock.
q-en-com.
q-différée.
q-pro-def.
poids-u.
prix-u.
u-mes.
i-qmin.
q-eco-com
ft-de-com
0-n
0-n
0-n
0-n
R-LIGNE-LIVR.-F
num-pro
num-B.L.F
q-liv.-F
q-ref-F
motif-ref
R-LIGNE-C-F
num-pro
num-com-F
q-com-F
q-due-F
DD-liv-F
1-n
1-n
BON-LIVR-F
num-B-F
date-livr-F
COM-FOURN
num-com-F
date-com-F
état-com-F
1-1
1-1
R-F-B-LIVR
num-B-F
num-F
R-F-COM-F
num-F
num-com-F
R-PRO-FOURN
num-pro
code-P.F
num-F
libellé-P.F
...
cofac-P.F
C.I.F 3
C.I.F 4
0-n
0-n
FOURNISSEUR
num-F
nom-F
adr-F
0-n

fig 7b.

•M.C.T. : modèle de traitements.

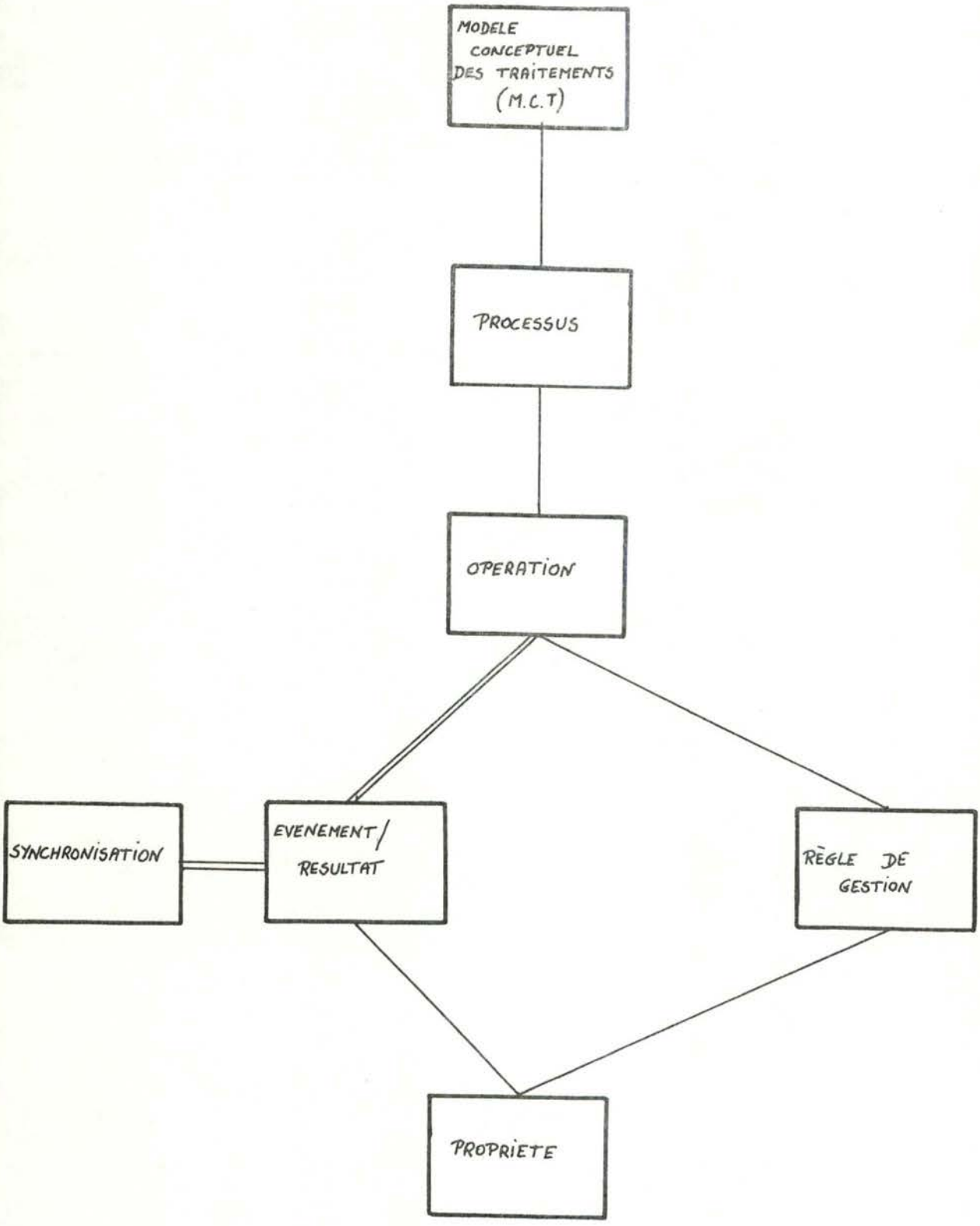
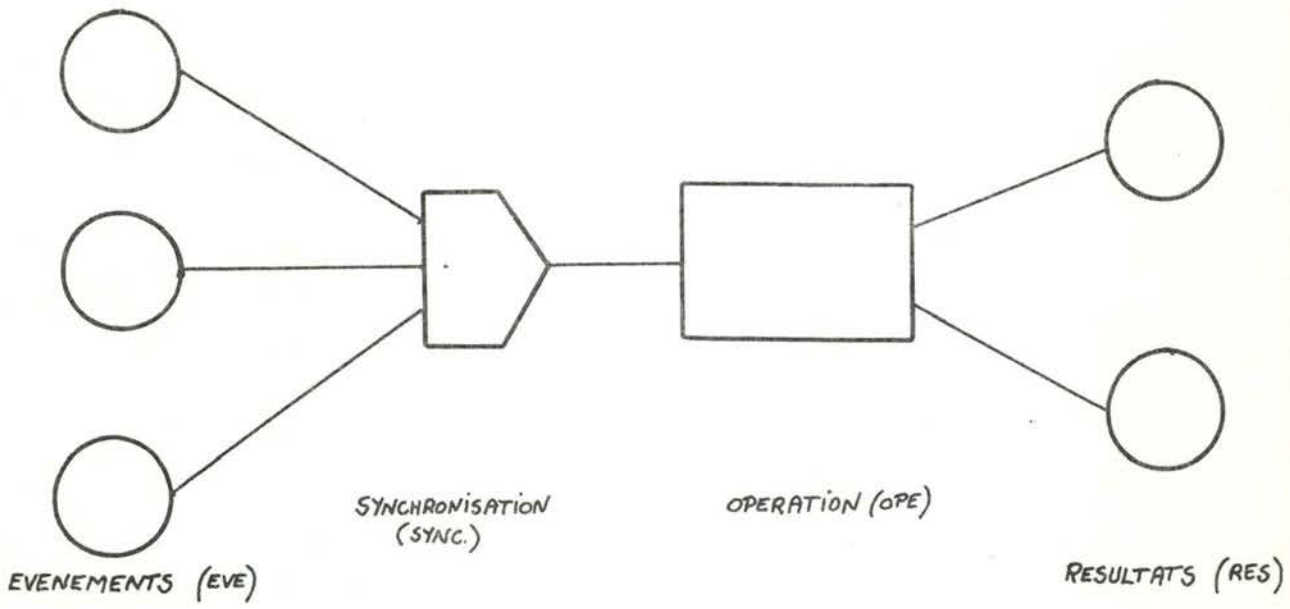


fig 8

Le formalisme défini pour exprimer le niveau conceptuel des traitements s'appuie sur 4 concepts :

- l'événement,
- la synchronisation,
- le résultat,
- l'opération,

concepts qui s'articulent de la façon suivante :



un ensemble d'événements déclenche, par l'intermédiaire d'une synchronisation, une opération qui produira un ensemble de résultats. Définissons de façon plus précise ces concepts :

L'opération :

-C'est la représentation d'un ensemble de traitements effectués par les S.I. en réaction à un stimulus. Une fois déclenchée, l'opération prend en compte les données provenant des événements en entrée; elle consulte et/ou met à jour les données mémorisées dans la B.D. et prépare un ou plusieurs résultats en sortie. Ces résultats peuvent être obligatoires, facultatifs ou même exclusifs en fonction d'un prédicat conditionnant leur émission et il faut également signaler qu'au cours de ses activations

successives, une opération n'exécutera pas nécessairement les mêmes traitements et pourra donc produire des résultats différents.

-Elle se déroule sans attente d'aucun événement complémentaire et ne peut être interrompue.

On peut lui associer une durée.

Rmq. : au niveau conceptuel, on fait l'hypothèse suivante : l'allocation complète des ressources (données + processeurs) avant le déroulement de l'opération.

L'événement.

C'est la représentation d'un fait nouveau pour le S.I .

Il présente deux aspects :

- c'est un stimulus qui franchit à un instant donné la frontière du S.I et y provoque une réaction;
- c'est un message c'est-à-dire un ensemble de données associées au fait à prendre en considération.

On distingue :

- l'événement externe : événement associé à un fait qui seproduit à l'extérieur du S.I. (soit dans le S.P., Le S.O. ou l'environnement de l'organisation) (1)
- l'événement interne : événement associé à un fait interne au S.I.

•Description d'un événement-type :

Un événement-type se caractérise par :

- un nom,
- une date de réception dans le S.I.,
- la description du fait nouveau :
  - identifiant(s) du (ou des) objet(s) mis en cause,

---

(1) cfr. Introduction.

- propriétés associées au fait nouveau (temps, lieu, circonstances,...) .

- et éventuellement des attributs complémentaires concernant :

  - son origine,
  - sa transmission (canal, code) ,
  - sa réception dans l'organisation et dans le S.I.,
  - sa durée de vie : durée au-delà de laquelle l'occurence de l'événement disparaît (si elle n'est pas précisée, elle est considérée comme illimitée) ,
  - sa capacité : nombre maximal d'occurences d'un même événement-type que l'on peut admettre à un instant donné dans le S.I.

·Description d'une occurence d'événement :

Deux occurences d'un même événement-type diffèrent l'une de l'autre par les valeurs prises par les propriétés de l'événement-type et par leur instant de réception par le S.I.
Si l'événement, après traitement, doit être mémorisé sous forme d'individu,il est alors nécessaire de munir l'événement-type d'un identifiant unique qui deviendra l'identifiant de l'individu-type : ex. :

- événement : arrivée d'une commande
  → individu : commande , dont l'identifiant sera : n° cde .

l'individu reprendra également les valeurs correspondant aux propriétés de l'événement-type .

Rmq. : à chaque événement est associé une variable d'état qui permettra de contrôler voire simuler le fonctionnement général du modèle (1).

---

(1) cfr. chap.V.1.6.

Le résultat:

C'est la représentation de la réponse codifiée du S.I. produite par une opération.
On distingue :

- le résultat externe : résultat qui sort du S.I. vers l'environnement, le S.P. (2) ou le S.O. (2) .
- le résultat interne : ce résultat ne sort pas du S.I. vis-à-vis duquel il pourra à son tour jouer le rôle d'événement (interne en l'occurence) ; ainsi, un résultat interne peut lui aussi participer à l'activation de nouvelles opérations.

Rappelons que l'émission d'un résultat par une opération peut être conditionnée par une règle de gestion (cfr. ci-après) particulière (règle d'émission faisant intervenir des propriétés des événements /résultats internes en entrée de l'opération).

Rmq. :
- les notions de résultat-type, d'occurence de résultat et leur description sont définies de façon analogue à celles des événements.
- à chaque résultat est associé une variable d'état qui permettra de contrôler voire simuler le fonctionnement général du système. (1)

La synchronisation :

C'est une précondition pour l'activation d'une opération à partir de plusieurs événements/résultats.
La synchronisation est spécifiée par le nom des différents événements et/ou résultats qui y contribuent, et par un prédicat qui précise comment ces événements/résultats participent à la synchronisation.

---

(1) cfr. chap.V.1.6.
(2) cfr. Introduction.

Tant que le prédicat associé à la synchronisation n'est pas vérifié, celle-ci reste en attente, ainsi que les occurences présentes sauf si une durée de vie, affectée dès le départ à l'occurence, est dépassée. Par contre, dès que le prédicat est vérifié, la synchronisation produit à sa sortie un événement interne unique porteur de l'ensemble des propriétés des événements et/ou résultats internes en entrée, sans transformation de ces dernières; cet événement interne déclenche alors l'opération.

Rmq. : - le prédicat associé à la synchronisation ne porte pas sur les données mémorisées; il a pour opérateurs : ET, OU exclusif, NON, (,) .

- plusieurs occurences des événements/résultats pouvant être présentes simultanément, on précise celles qui doivent être effectivement choisies au moyen d'une ou plusieurs condition(s) locale(s) , condition(s) qui porte(nt) sur les valeurs des propriétés des événements/résultats à synchroniser.
  Lorsqu'une condition locale porte uniquement sur un identifiant commun, elle reste souvent implicite.

- la synchronisation peut être pilotée par un moniteur qui soumettra son déclenchement à une variable d'état majeure (1).

A ces quatre concepts principaux s'ajoutent :- le processus,
- la règle de gestion,
respectivement en amont et en aval de la notion d'opération.

Le processus :

Sous-ensemble du M.C.T., c'est un ensemble connexe d'opérations, synchronisations, événements et résultats qui concourent à un même but et que l'on décide de regrouper de ce fait.

Rmq. : à un processus, on associera généralement un modèle externe de données (M.E.D.) ; on peut donc considérer le processus comme un modèle externe de traitement (M.E.T.).

---

(1) cfr. chap.V.1.6.

La règle de gestion :

C'est une unité de description, à un niveau conceptuel, de l'action exercée sur les données par le système.
Au niveau conceptuel, on peut se limiter à l'énoncé de(s) la R.E.G. et/ou de(s) la règle de calcul (R.E.C.) en explicitant dans les grandes lignes leur action respective et en citant les propriétés en entrée et en sortie pour chacune de ces règles.

ex. : Règle 1 : "Définition d'une quantité à approvisionner en fonction d'un seuil et d'une quantité économique"

La propriété :

Sa définition est celle fournie lors de la présentation du M.C.D.
Elle intervient cependant ici encore :

- dans la définition des types d'événements/résultats.
  (cfr. : 'Description d'un événement-type' p.)
- dans la spécification des règles de gestion.

Exemple de M.C.T. MERISE.

Nous allons maintenant construire le M.C.T. MERISE du cas "PETITPAS" et identifier ainsi les opérations, événements, résultats et synchronisations qui y participent. Nous nous contenterons ici d'étudier le processus "Traitements des commandes clients".

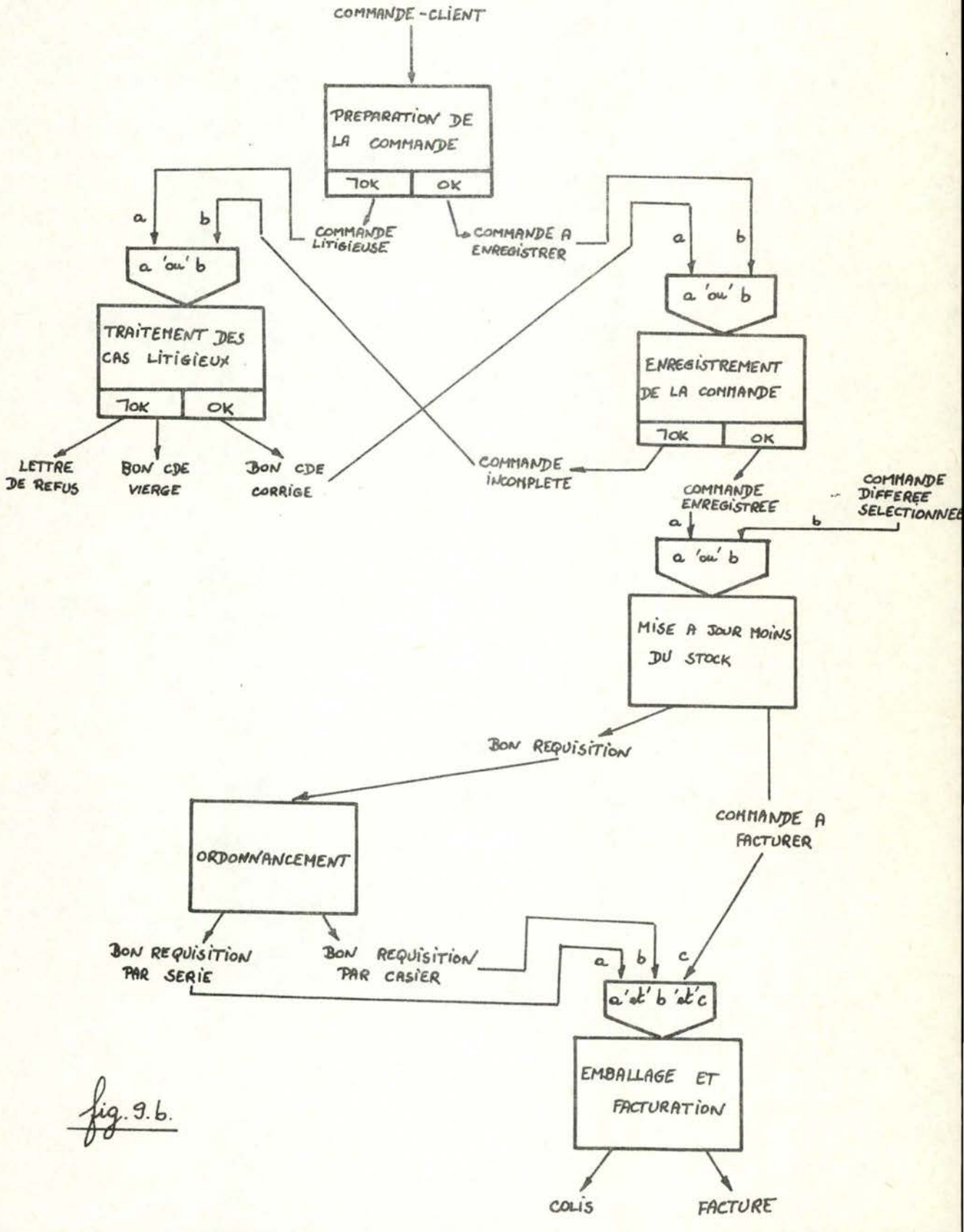


fig. 9.b.

Après avoir notamment identifier les opérations du M.C.T., nous chercherons les règles de gestion (R.E.G.) associées. Nous nous contenterons ici de reprendre une opération et de donner sa liste de R.E.G. (1)

Ex. : Opération : Préparation de la commande

R.E.G. relative à la vérification de l'identité du client :

1. Tout client est identifié soit par son numéro de client, soit par ses nom et prénom.
2. Toute commande émanant d'un tiers non client, provoque une adjonction à la collection des clients. Ce tiers devient client et reçoit un numéro de client attribué par compostage.
3. Les nom, prénom, adresse inscrits par le client sur son bon de commande ont toujours priorité sur le numéro inscrit par le client sur ce bon de commande lors d'un essai d'identification.
4. Il n'y a pas de numéro de client sur le bon de commande et les nom, prénom, adresse ne figurent pas non plus sur ce bon de commande. Alors, on refuse la commande.
5. ...

---

(1) Comme nous le verrons plus tard, on peut assimiler les R.E.G. MERISE aux règles de traitement de 'NAMUR'.

II 1.2. 'NAMUR' : modèles de niveau conceptuel.

Modèle des données :

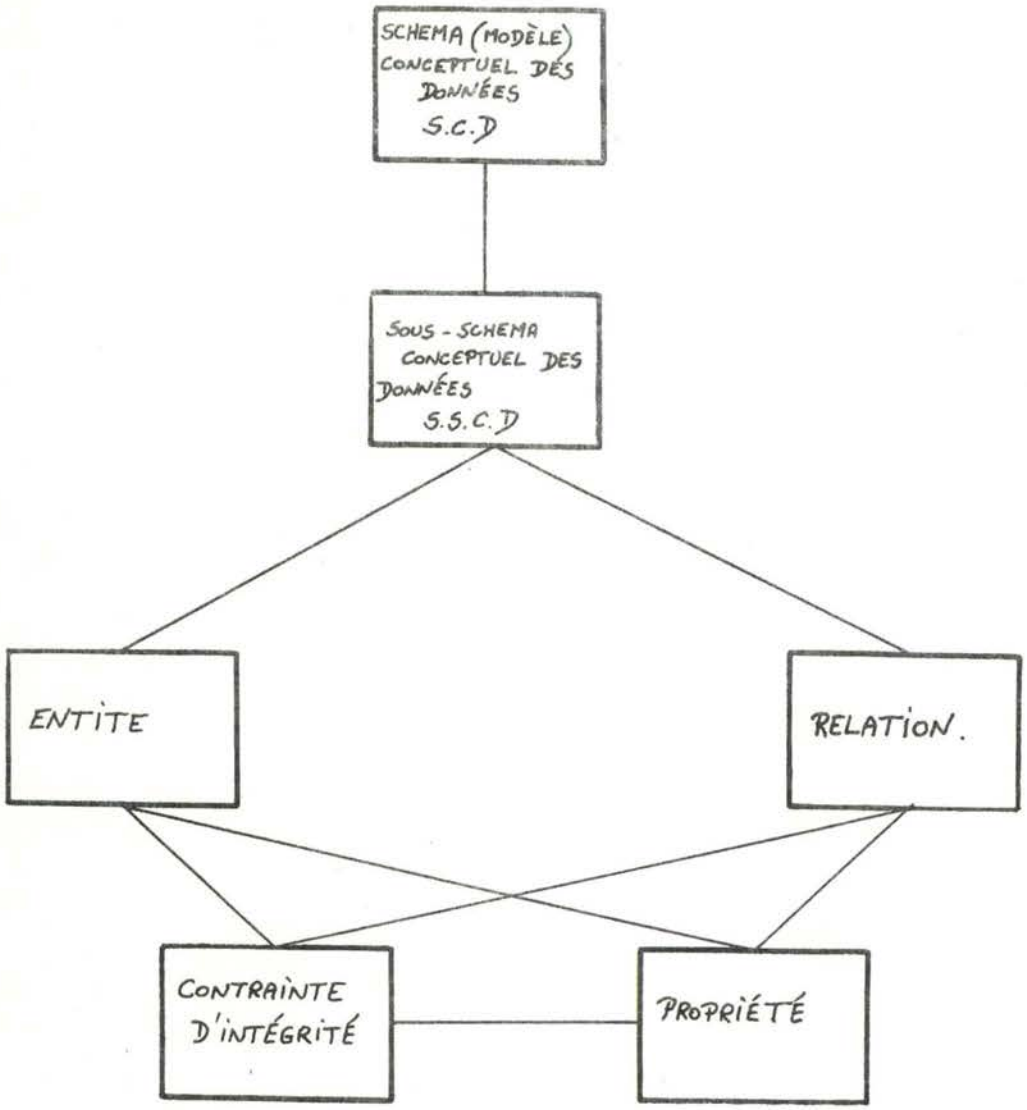


fig 10.

Les concepts fondamentaux sont ici ceux d'entité, d'association et de propriété,[1] concepts très proches des individu, relation et propriété de MERISE.

---

(1) "Problèmes d'organisation et méthodes d'analyse fonctionnelle" F.BODART (1979).

L'entité :

-Une entité est ce qu'un individu ou un groupe d'individus voit comme un tout ayant une existence propre.
-Elle est caractérisée par un ensemble de propriétés qualitatives et quantitatives et un comportement permanent.
-Un type d'entité est un ensemble d'entités caractérisées par les mêmes propriétés; l'occurence d'entité (une entité) étant quant à elle une réalisation possible de ce type d'entité. (1)
Ex. : un client, une commande, un véhicule, ...

L'association :

-C'est un ensemble de deux ou plusieurs entités où chacune assume un rôle donné.
-Elle peut posséder plusieurs propriétés et n'a d'existence que par rapport aux entités qu'elle relie.
-Un type d'association est un ensemble d'associations caractérisées par les mêmes propriétés; une occurence d'association (une association) étant quant à elle une réalisation possible de ce type d'association.
Ex. : association "propriétaire" entre les entités "personne" et "véhicule",
association "ligne de commande" entre les entités "commande" et "produit",...

La propriété :

-C'est une qualité que l'on attribue à un type d'entité ou d'association ou, plus formellement, c'est une relation entre :

{ - une entité ou une association
{ - une ou plusieurs valeurs qui la caractérisent.

Ex. : nom d'un individu (propriété d'un type d'entité) ,
n° de carte grise d'un propriétaire de véhicule (propriété d'un type d'association) ,...

---

(1) Rappel : chaque niveau de description est réalisé essentiellement en termes de types; les occurrences ne sont prises normalement en charge qu'au stade d'exécution.

On peut regrouper les propriétés en trois grands groupes :

- celles qui identifient l'entité,
- celles qui caractérisent l'usage de l'entité ou de l'association au sein de l'organisation,
- celles qui localisent dans le temps et dans l'espace les propriétés d'usage.

Rmq. : signalons qu'une propriété peut, à un moment donné, prendre la valeur 'Inconnue' ou 'Indéterminée'.

Les contraintes d'intégrité :

Propositions logiques définies sur les entités, les associations ou les propriétés, elles doivent être vérifiées à tout moment afin de respecter le comportement du réel perçu.
On distinguera souvent dans l'expression d'une contrainte d'intégrité :

- le domaine contraint,
- le domaine contraignant,
- le prédicat,
- les conditions d'activation.

Ces contraintes d'intégrité portent principalement :

- sur les formats des propriétés:

  Ex. : montant à payer : FORMAT '9 (6)' ,

- sur la cardinalité d'une entité:

  Ex. : client : CARDINALITE 10.000 (il existe 10.000 clients pour la firme considérée) ,

- sur la cardinalité d'une association:

  Ex. : ligne-de-commande-client : CARDINALITE : 200.000 (on compte 200.000 lignes de commande pour les clients).

- sur les connectivités des associations :

  Ex. : association client/commande-client : CONNECTIVITE : 1-n, 1-1

- sur les valeurs des propriétés d'une entité ou d'une association.

Ex. : pour une ligne- de- commande-client, la quantité livrée est la quantité d'un produit envoyée par la firme au client lors de toute livraison

- domaine contraint : q-livrée-cli
- domaine contraignant : q-cdée-cli
- prédicat : q-livrée-cli $\leqslant$ q-cdée-cli
- condition d'activation : lors de la création, on égale q-livrée-cli à 0.

Ou encore : pour une unité de mesure du produit,

table de codification : 01 : pièce
02 : paire
03 : mètre courant en 1,40 m.
04 : mètre courant en 0,60 m.

- sur l'existence d'une entité, d'une association ou d'une propriété.

Ex. : l'association client/prospect ne peut exister que s'il existe pour le client associé au prospect donné au moins une occurrence de l'association client/commande-client.

- domaine contraint : association client/prospect
- domaine contraignant : association client/commande-client
- Prédicat : n'existe que s'il y a au moins 1 occurrence du domaine contraignant.
- condition d'activation : lors de la création de l'association client/prospect.

Sous-schéma conceptuel-schéma conceptuel.

Un sous-schéma est une structure de données composée de types d'entités et d'associations, de propriétés, et de contraintes d'intégrité. Il est propre à la réalisation d'un traitement homogène, application ou phase (1) .

Le schéma conceptuel intègre les différents sous-schémas en une structure globale, représentation exhaustive et non redondante du réel perçu de l'organisation.

C'est un niveau d'accord entre les informaticiens et les utilisateurs, un cahier des charges en vue de l'implémentation, un invariant par rapport aux niveaux logique et physique.

La représentation graphique de structures de données est semblable à celle utilisée par MERISE . (p.21)

(1) Cfr. ci-après : modèle conceptuel des traitements.

Exemple de S.C.D. 'NAMUR'.

Reprenons le cas "PETITPAS" (1) et élaborons le S.C.D. (cfr. p.39). Le S.C.D. ne serait pas complet si nous n'y ajoutions pas un ensemble de contraintes d'intégrité (cfr. p.35). Il serait trop long de les citer toutes ici, contentons-nous d'en donner quelques exemples :

- Relation r - ligne - exp;
  - Contrainte d'intégrité : la valeur de n^* dans la connectivité de r - ligne - exp ne peut être supérieure à celle de n^* dans la connectivité de r - ligne - cc.
  - Type de contrainte : valeur.
  - Domaine contraint : connectivité de l'association r - ligne - exp.
  - Domaine contraignant : connectivité de r - ligne - cc.
  - Prédicat : voir contrainte d'intégrité.
  - Condition d'activation : lors de la création.
- Elément Adr - cli:
  - Format 'x (60)';
- Connectivité de l'association r - com - cli : 1-n, 1-1;
- Elément date - cc :
  - Table de codification :
    - caractères 1-2 : JJ (jour du mois)
    - caractères 3-4 : MM (mois de l'année)
    - caractères 5-6 : AA (2 derniers chiffres de l'année),
- ...

---

(1) du moins la version simplifiée retenue par J.L.HAINAUT dans "Analyse du cas "PETITPAS" - Dossier d'analyse organique - Première partie : la base de données".

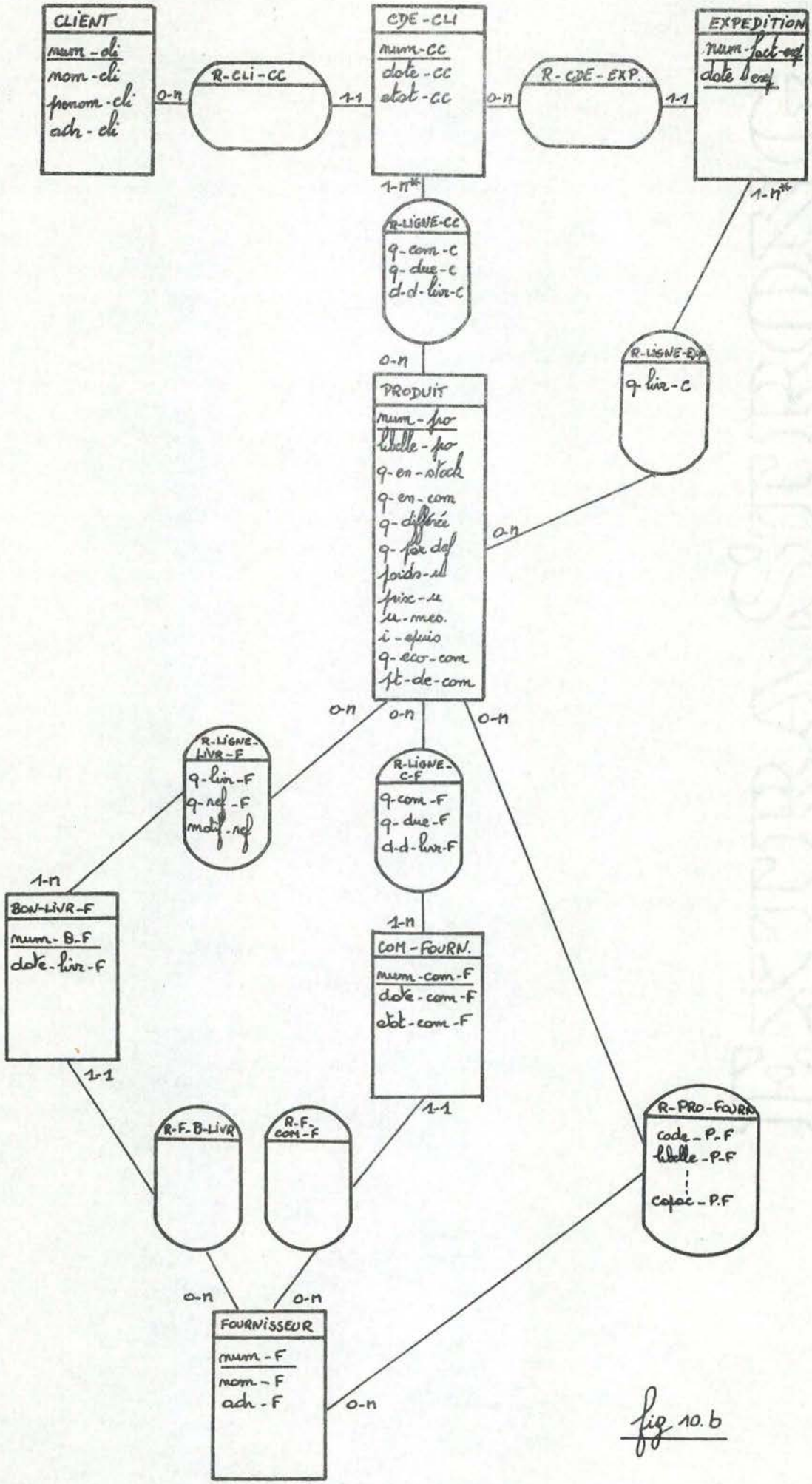

CLIENT
num-cli
nom-cli
prenom-cli
adr-cli
0-n
R-CLI-CC
1-1
CDE-CLI
num-CC
date-cc
etat-cc
0-n
R-CDE-EXP.
1-1
EXPEDITION
num-fact-exp
date exp
1-n*
1-n*
R-LIGNE-CC
q-com-c
q-due-c
d-d-liv-c
0-n
R-LIGNE-EXP
q-liv-c
PRODUIT
num-pro
libelle-pro
q-en-stock
q-en-com
q-différée
q-pre def
poids-u
prix-u
u-mes.
i-qmin
q-eco-com
pt-de-com
0-n
0-n
0-n
R-LIGNE-LIVR-F
q-liv-F
q-ref-F
motif-ref
R-LIGNE-C-F
q-com-F
q-due-F
d-d-liv-F
1-n
BON-LIVR-F
num-B-F
date-liv-F
1-1
1-n
COM-FOURN.
num-com-F
date-com-F
etat-com-F
1-1
R-F-B-LIVR
R-F-COM-F
R-PRO-FOURN
code-P-F
libelle-P-F
copac-P-F
0-n
0-n
FOURNISSEUR
num-F
nom-F
adr-F
0-n

fig 10.b

Modèles des traitements.

Nous aborderons la modèlisation des traitements en trois phases : (1)

a) décomposition des traitements en fonction d'une nomenclature donnée,
b) description statique,
c) description dynamique ;

a) Elements d'une nomenclature hiérarchique pour la décomposition des traitements.

Remarque préalable.

La nomenclature hiérarchique des traitements que nous allons proposer ici, n'est pas exclusivement de nature conceptuelle : elle sera, en effet, reprise telle quelle au niveau logique des traitements où l'on explicitera cependant les fonctions en fournissant un ensemble d'algorithmes qui mettront en évidence les accès et les manipulations sur une structure de données (nous aurons l'occasion de revenir sur ce point.).
Quant au niveau physique des traitements, il se base sur une autre nomenclature dont la notion principale est celle de 'programme'.

Venons en maintenant à la description des éléments de la nomenclature que nous avons retenue :

La fonction :

-Correspondant au niveau élémentaire des traitements, elle reflète le degré de modularité choisi pour la programmation.
-Comme critères d'individualisation, on reconnaît généralement :

- qu'une fonction doit avoir une sémantique propre,
- qu'une fonction automatisable ne peut posséder qu'un seul point d'entrée et un seul point de sortie,

---

( )"Modèles et outils d'aide à l'analyse des spécifications d'un système d'information". F.BODART AFCET-80.



fig 11

- qu'une fonction programmable devrait pouvoir être réalisée par un module de programmation (unité compilable).

C'est un élément d'une phase.

La phase :

Elle peut être décrite comme un graphe sans circuit dont les sommets sont les fonctions et les arcs, les relations de succession entre les fonctions.

C'est un traitement possédant une unité spatio-temporelle d'exécution.

Elle est caractérisée entre autres par :

- une homogénéité des ressources associées à son exécution,
- la succession logique vis-à-vis des phases dont l'exécution est asynchrone et la terminaison, la condition de son déclenchement,
- la dépendance logique de plusieurs phases dont l'exécution est asynchrone et le déclenchement conditonné par la terminaison de la phase considérée,
- l'interruption du traitement de l'information par un phénomène externe (prise de décission, contrôle,..
- le fonctionnement asynchrone de phases (différence de périodicité dans l'exécution, fréquence distincte d'acquisition des inputs,...)

Une phase sera exécutée dans le cadre d'une cellule d'activité, centre d'activité homogène dansle temps et dans l'espace pourvu de ressources et de règles de comportement nécessaires à son fonctionnement.

L'application :

Décrivant l'enchainement des phases relatives à un flux d'informations, elle peut être représentée par un graphe dont les sommets sont les phases et les arcs, les relations de succession entre les phases.
Elle a pour caractéristiques :

- une existence quasi-autonome ne communiquant avec les autres applications que par des agrégats d'informations situationnelles qui représentent des propriétés d'état,
- une continuité dans le temps d'un ensemble d'activités,
- son association avec un flux homogène d'informations pour lequel il existe des inputs et des outputs bien définis.

Sous-système.Système :

C'est un ensemble d'applications - d'exécution (appartenant aux flux physiques)

- de gestion (appartenant aux flux de décision)

qui définissent et contrôlent le comportement des premières.

Les différents sous-systèmes sont alors regroupés pour former le système de l'organisation.

b) Description statique des traitements.

Elle consiste en une spécification des éléments en entrée et en sortie du traitement considéré : application, phase ou fonction.

Ces éléments correspondent à des messages (un MESSAGE est une transaction qui est soit générée (le message est alors un 'output'), soit reçue (le message est alors un 'input') par le système considéré vers ou en provenance de son environnement, ou c'est une transaction qui circule à l'intérieur du système, d'u processus à un autre (dans ce cas, le MESSAGE est une 'transaction' entre processus)) (1) et à des collections d'informations.

Ainsi nous avons :

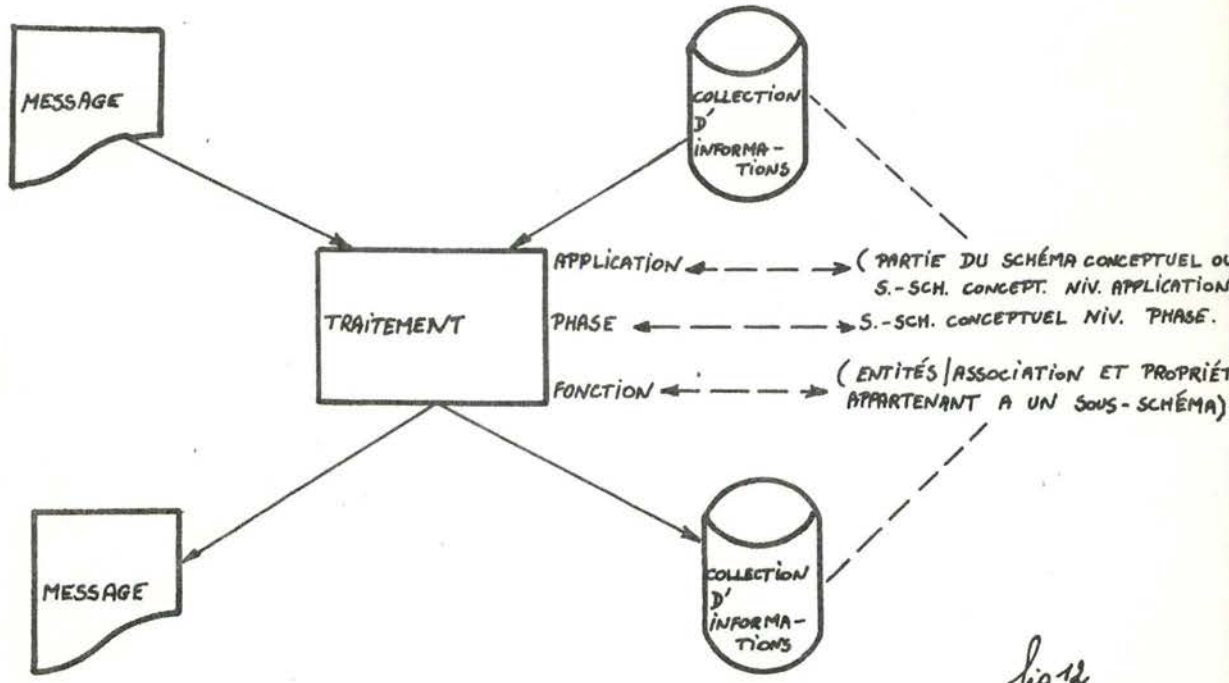


fig 12

(1) "D.S.L. Users'Manual" NAMUR version 01 F.BODART-Y.PIGNEUR.

On complète alors la description statique par l'énumération d'un certain nombre de règles spécifiant l'(es) action(s) à effectuer par les traitements de type fonction : on décrit le QUE FAIRE . Ces règles sont associées à des conditions données mais ne sont pas mises sous forme d'algorithmes : réalisations particulières d'un ensemble de règles et que l'on ne détaillera qu'au niveau logique.

c) Modélisation de la dynamique des traitements.

Trois concepts fondamentaux sont nécessaires pour exprimer la dynamique des traitements ; ce sont ceux de :
- processus
- événement
- point de synchronisatio

Le processus :

-Traitement d'un certain type, application-phase-fonction, dont une occurrence est créée à chaque exécution de sa PROCEDURE : attribut d'un processus décrivant son action comme un ensemble de règles et d'actions reliées logiquement entre elles.(1)
-Du point de vue dynamique, la seule chose que l'on connaît d' un processus, c'est une estimation de son temps d'exécution, sa durée.En effet, on considère à ce niveau le processus comme une"boîte noire" et ses propriétés internes ne nous intéressent pas ici.
-Le comportement dynamique d'un processus ne sera décrit qu'en fonction de caractéristiques observables (externes), notamment celles relatives à son cycle de vie.
Ainsi, on retiendra les changements d'état suivants pour un processus :

activation : un processus déclenché et ayant les ressources nécessaires devient actif.

terminaison : la durée du processus étant écoulée, il est supposé terminé.

(1) "D.S.L. Users'Manual" NAMUR version 01 F.BODART - Y.PIGNEUR.

suppression : un processus peut être supprimé par une cause externe avant même la fin de sa durée estimée.

reprise : un processus interrompu peut redevenir actif grâce à un mécanisme externe.

interruption : un processus actif peut être interrompu par une cause externe.

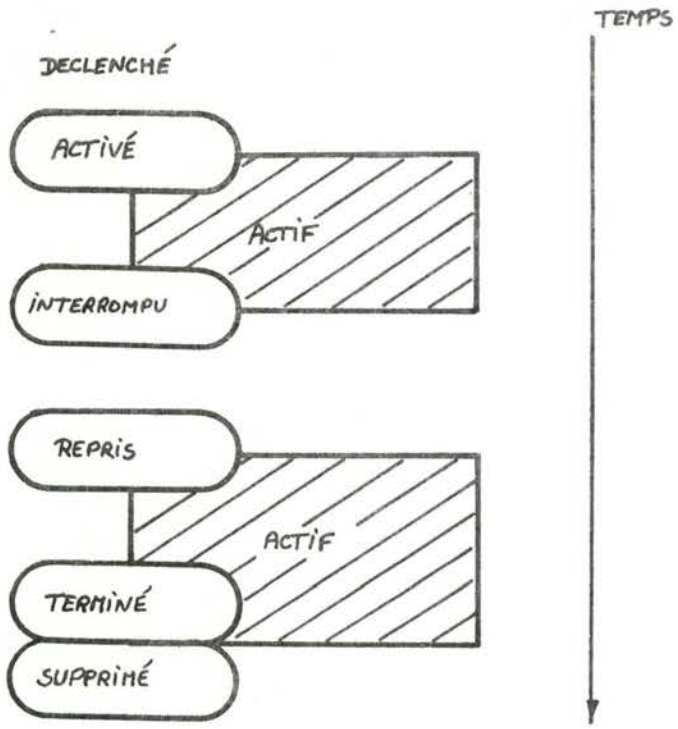


fig 13.

Tous ces changements d'état du processus sont générateurs d'événements.

L'événement :

Un événement est provoqué par un changement d'état du système et est connu d'un observateur uniquement par l'intermédiaire d'un message.
Le contenu du message associé à cet événement est spécifié en tant qu'attribut de l'événement (il y a notamment comme attributs, des précisions sur le temps et le lieu de génération de l'événement).
On distingue :

- l'événement externe : c'est un stimulus provenant

de l'environnement du système auquel ce dernier doit réagir.

- l'événement interne : événement qui se déroule à l'intérieur du système considéré, conséquence soit d'un changement d'état d'un processus, soit de l'émission d'un message interne.

On suppose que l'événement est le seul fait qui peut provoquer un changement d'état de processus. On peut donc compléter ainsi la figure 13 :

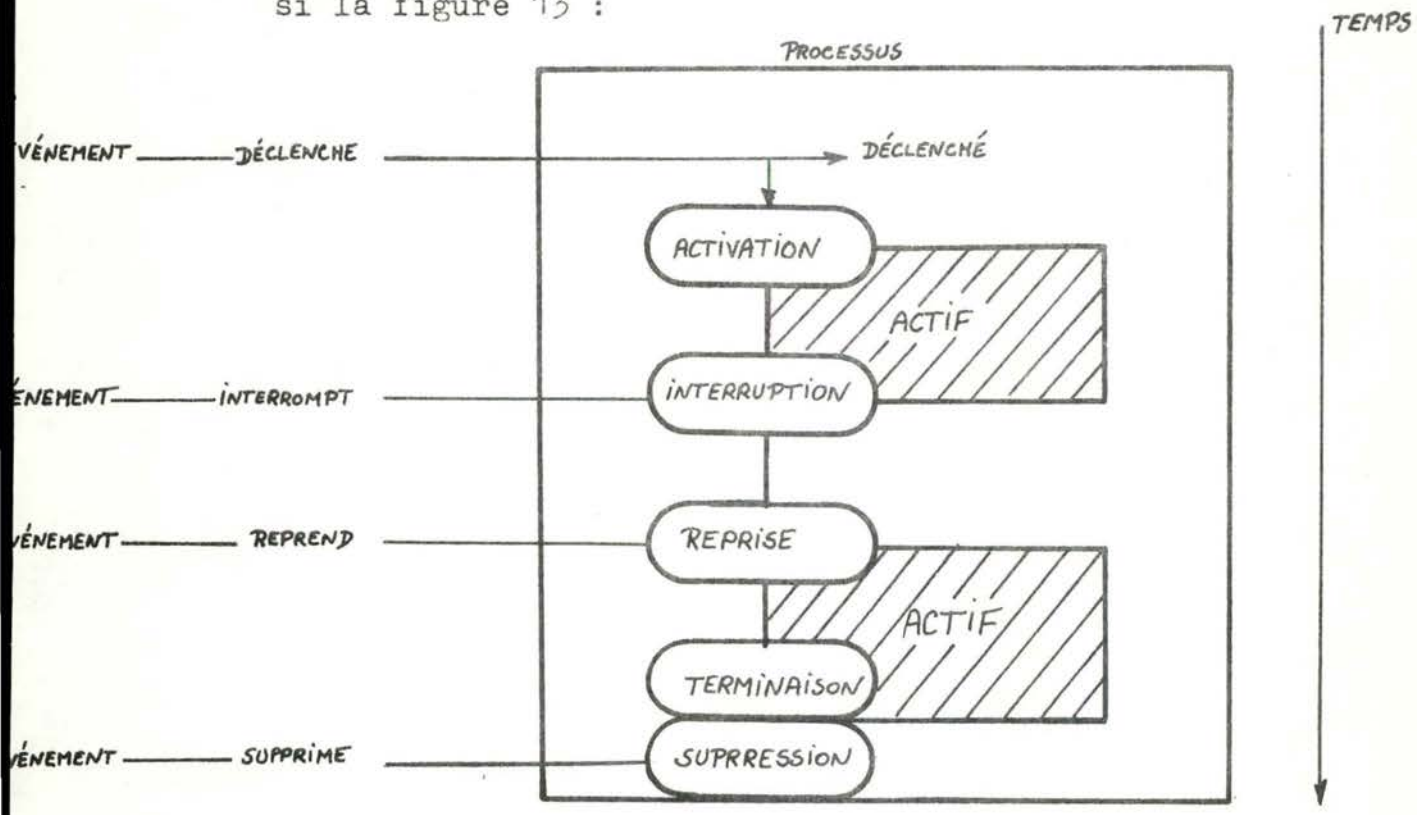


fig 14.

Une deuxième hypothèse que l'on fait également, c'est que l'événement doit être complet par lui-même, c'est-à-dire qu'il doit contenir tout ce qui lui est nécessaire pour spécifier les relations dynamiques qui lui sont associées et ne doit jamais dépendre de spécifications d'événements antérieurs.

Le point de synchronisation :

- C'est un point d'attente d'une combinaison d'un certain nombre d'événements.
  La liste des événements contribuant au point de synchronisation est donnée ainsi qu'un prédicat définissant :
  - la combinaison des événements entrants,
  - certaines conditions locales qui permettent de faire la discrimination entre plusieurs occurrences d'un même événement (ex. : événement arrivée d'une commande dont on peut différencier les occurrences grâce au n°de cde).
- Chaque réalisation d'un point de synchronisation génère une occurrence d'un certain événement (qui peut être implicite d'ailleurs) ; ce dernier a comme attributs implicites les attributs des événements contribuants.
- Par définition, chaque flux menant au point de synchronisation disparaît en tant que tel et est fondu dans un nouveau flux lors de la réalisation de cette synchronisation.
- La contribution d'un élément d'entrée à la synchronisation peut être limitée en durée : elle peut être instantanée, elle est effectuée jusqu'à la réalisation du point de synchronisation ou elle peut être limitée par l'émission d'un certain événement (c'est l'intérêt de la notion "durée de vie" d'un événement/résultat dans MERISE).

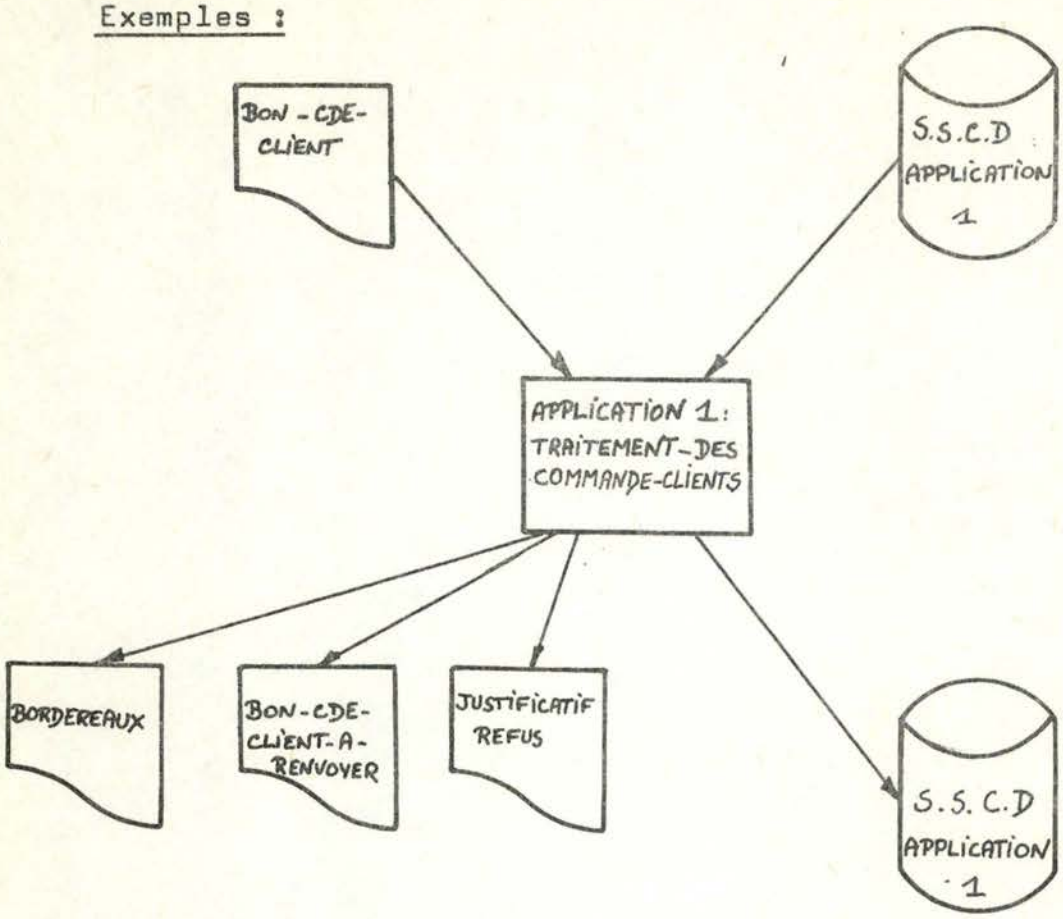
Exemple de modélisation conceptuelle des traitements : 'NAMUR'

Nous reprendrons ici les 3 phases de la modélisation conceptuelle des traitements : a) décomposition hiérarchique des traitements en fonction de la nomenclature donnée,
b) description statique,
c) description dynamique,
et nous l'appliquerons au cas "PETITPAS" (limité au traitement des commandes clients).

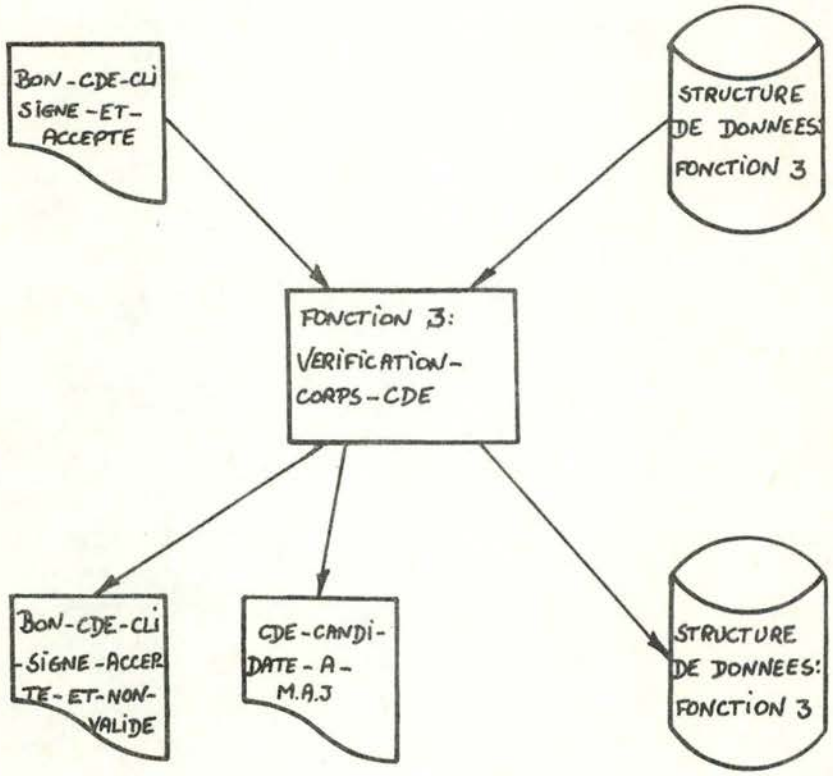
a) 1. Traitement-des-commandes-des-clients-et-des-fournisseurs : sous - système
2. Traitement-des-commandes-clients : application
3. Préparation-bon-cde-client : phase
4. Vérification-signature : fonction
3. Enregistrement-bon-cde-cli : phase
4. Vérification-ident-cli : fonction
4. Vérification-corps-cde : fonction
3. Traitement-des-cas-litigieux : phase
4. Correction-bon-cde : fonction
4. Refus-bon-de-cde : fonction
3. Mise-à-jour-moins-du-stock : phase
4. Maj-moins-stock : fonction
3. Ordonnancement-parcours-rayon : phase
4. Constitution-d'1-série : fonction
4. Ordonnancement : fonction
4. Déstockage : fonction
3. Constitution-du-colis : phase
4. Affichage-expédition : fonction
4. Reconstitution-cde : fonction
4. Facturation : fonction
4. Parcours-magasin-correct : fonction
4. Affichage-expéd-correct : fonction
4. Reconstitution-cde-corr : fonction
4. Correction-du-système : fonction
4. Emballage-et-expédition : Fonction
3. Sélection-cdes-différées : phase
4. Sélection-cdes-différées : fonction

b) Description statique :

Exemples :



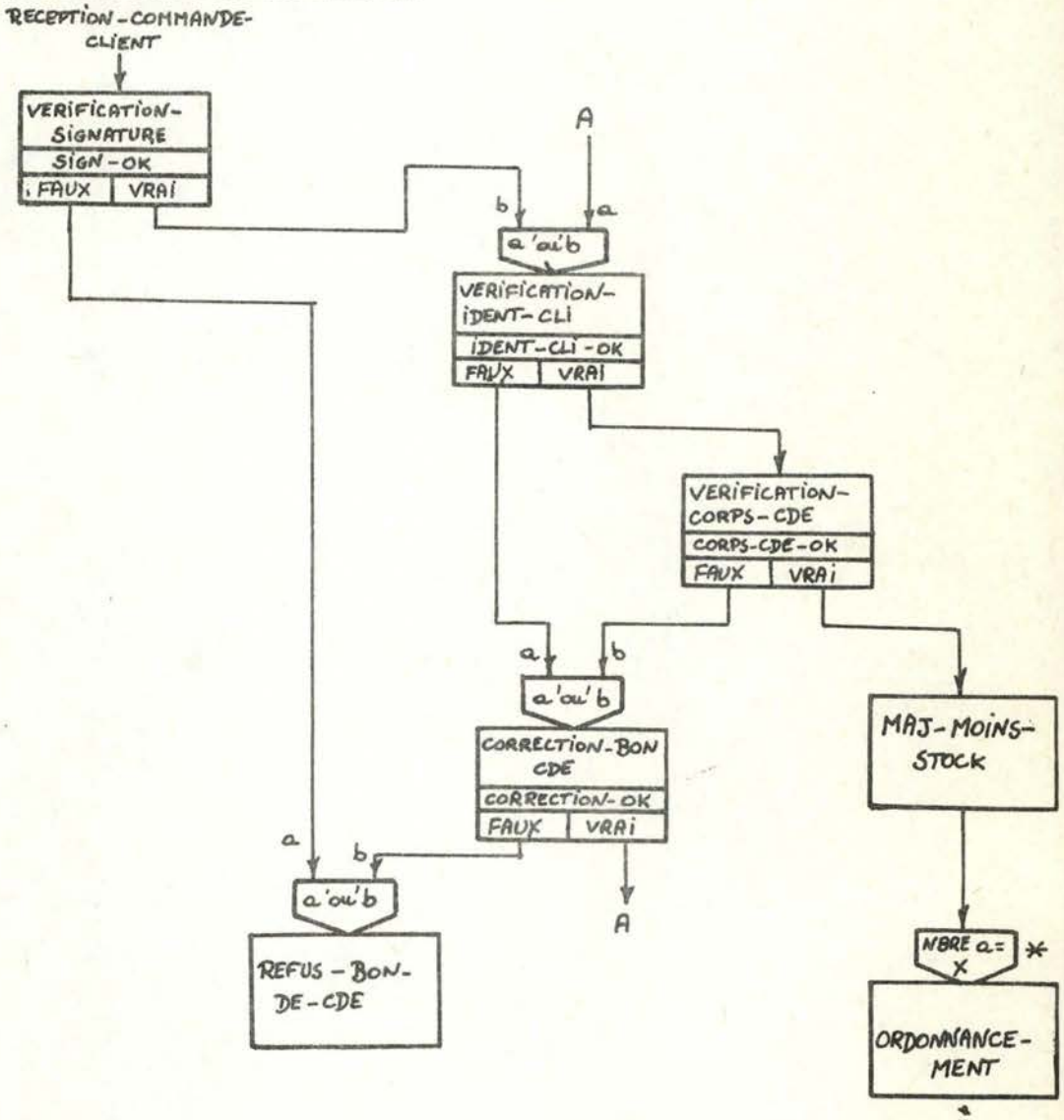
ou :



Exemple de règles de traitement : fonction : vérification-corps commande :

1. Toute ligne de commande est identifiée soit par un numéro de produit, soit par un libellé de produit.
2. Le libellé d'un produit a prépondérance sur le numéro lors d'un essai d'identification.
3. Une ligne de commande est acceptée si elle a pu être identifiée.
4. Une commande est acceptée si toutes ses lignes sont acceptées et si le montant des articles inscrit par le client est validé par l'opératrice.
5. ...

c) Description de la dynamique.



: point d'attente jusqu'à ce que le nombre de terminaisons du processus 'maj-moins-stock' soit égal à un nombre que l'on aura fixé.

II 1.3. Commentaires.

Nous venons donc de décrire les différents modèles de niveau conceptuel préconisés par les deux approches que nous étudions.
La comparaison s'avère ici assez facile en ce sens que les concepts et les règles retenues pour MERISE et 'NAMUR' sont fort semblables.

- Pour les données :

Nous sommes en présence de deux modèles qui utilisent les mêmes concepts, à savoir :
- l'entité-l'individu,
- l'association-la relation,
- la propriété,
- la contrainte d'intégrité.

Il subsiste cependant quelques petites différences :

a) pour MERISE, un type de propriété ne peut appartenir qu'à un seul individu-type ou à une seule relation-type dans le M.C.D. (convention de NAMING sur la qualification.).

b) pour MERISE, une relation-type doit avoir pour identifiant la concaténation des identifiants des individus-types qu'elle associe. Cela s'avèrera utile lors du passage au niveau logique, comme nous le verrons plus tard. 'NAMUR', qui utilise quant à elle une autre représentation au niveau logique, n'a pas besoin d'introduire cette contrainte de niveau conceptuel.

c) MERISE introduit aussi explicitement la notion de contrainte d'intégrité fonctionnelle qui sert à la décomposition des relations et à l'optimisation du M.C.D.

- Pour les traitements :

Au point de vue de la dynamique des traitements, on peut également dire que les concepts de base sont fort semblables à savoir que l'on a :

- l'opération-le processus,
- l'événement/le résultat-l'événement,
- la synchronisation-le point de synchronisation,

respectivement pour MERISE et pour 'NAMUR'.

Quelques remarques s'imposent cependant :

a) pour MERISE, l'opération est non-interruptible ; il faut donc pousser la décomposition des traitements pour qu'il en soit ainsi. Par contre, pour 'NAMUR', un processus peut être momentanément interrompu pour être repris par la suite (cfr. fig. 14 : cycle de vie d'un processus).

b) pour 'NAMUR', on propose déjà au niveau conceptuel la distinction en application-phase-fonction pour les processus alors que pour MERISE, seule la notion d'opération existe au niveau conceptuel. Une classification des opérations en procédure-phase-tâche (1) interviendra cependant au niveau logique grâce à un certain nombre de critères d'individualisation et à un ensemble de choix organisationnels. On peut tout de même déplorer le fait que la définition de l'opération n'est pas toujours suffisante pour permettre une découpe précise et cohérente des traitements.

c) MERISE distingue les événements des résultats bien qu'il n'y ait aucune différence formelle entre eux si ce n'est par rapport à un ensemble de traitements donné.
Pourquoi alors avoir introduit cette notion de résultat ?
Trois raisons sont avancées par H. HECKENROTH :

- historiquement, lors de l'élaboration des modèles de traitement, MERISE a conservé une distinction événement/résultat qui était déjà apparue dans certaines méthodes d'analyse (ex. : CORIG, DE BLAMPRE) ;
- de plus, cette distinction ne présentait aucun inconvénient pour le développement de certains outils (CHARCATI, AVALCATI ,...) (2) et était même très utile dans d'autres (ex. : COECATI) (2) ;
- un argument qui était également avancé, c'est la différence d'origine de l'événement et du résultat

---

(1) cfr. chap. V : outils développés dans le cadre des 2 méthodes.
(2) cfr. ci-après : modèle logique des traitements.

(l'événement interne, produit d'une synchronisation, n'étant qu'un artifice de construction) : le premier provient de l'extérieur du S.I. (environnement, système opérant (S.O.), système de pilotage (S.P.),); le second quant à lui est une "réponse codifiée du S.I. produite par une opération".

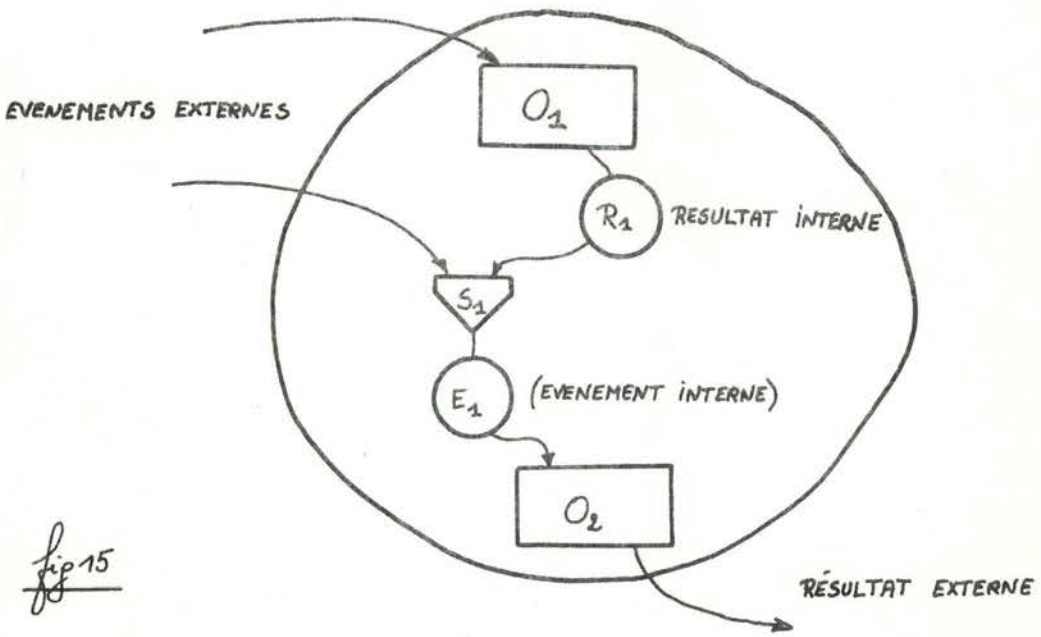


fig 15

Dans l'approche 'NAMUR', on n'a introduit qu'un seul concept : l'événement, qui peut déclencher un processus ou être produit par lui.

d) enfin, dans 'NAMUR', la description statique des traitements met l'accent sur les structures de données utilisées et éventuellement modifiées par les différents types de traitements ainsi que les entrées et les sorties propres à chacun d'eux (MESSAGES). Cela n'apparaît pas de façon explicite dans MERISE.

Pour conclure cette partie relative au niveau conceptuel, nous signalerons qu'il existe d'autres approches, différentes de celles que nous venons de voir, mais qui ne manquent pas d'intérêt.
Une d'entre elles est étudiée à Nancy (C.Rolland, S.Leiffert, C.Richard

et se base sur un modèle de type relationnel incluant le temps et utilisant des types pour définir un schéma conceptuel unique. Ce schéma englobe l'aspect statique (sous-schéma statique : ensemble de relations qui correspond à la structure des données et représente la structure des composants de l'organisation) et l'aspect dynamique (sous-schéma dynamique : ensemble de relations qui représente la dynamique de l'organisation par le réseau des interconnexions entre les catégories de phénomènes).

II 2. Modèles de niveau logique.

II 2.1. MERISE : modèles logiques.

M.L.D. : modèle logique des données

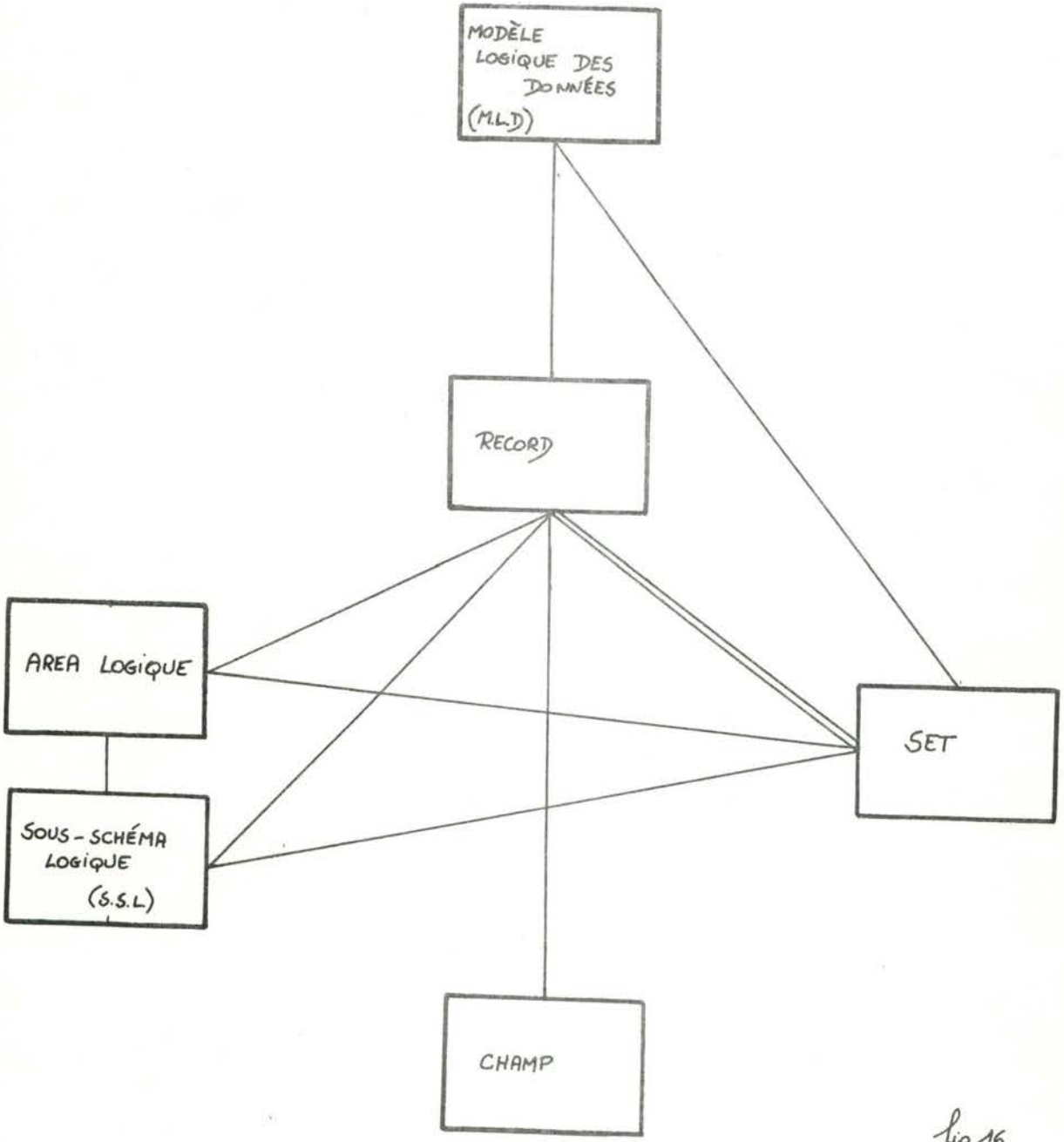


fig 16

Les différents concepts sont, à peu de choses près, ceux proposés par le Data Base Task Group CODASYL :

Type de champ :

- C'est la plus petite partie d'une donnée nommée.
- l'occurrence d'un type de champ est la représentation d'une valeur.

type de record :

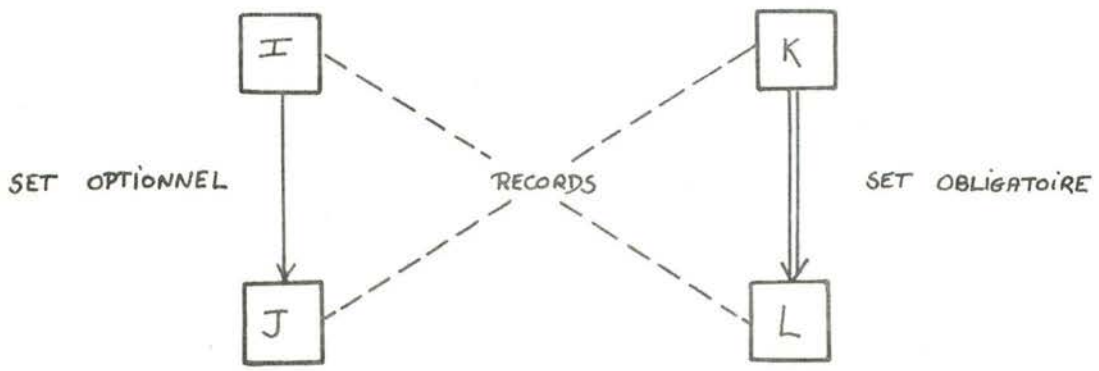
- c'est une collection nommée, sans répétition, de n types de champ ($n > 0$)
- clé d'un type de record : c'est un type de champ permettant d'identifier de façon unique une occurrence de type de record.

type de set :

- c'est une relation qualifiée entre un type de record déclaré comme "maître" et un type de record déclaré comme "membre".
- les occurrences d'un type de set se composent d'une occurrence de record "maître" et de n occurrences de record "membre" ($n \geqslant 0$).
- cette notion de set assure une liaison entre les records et permet l'expression de structures d'informations.
- on distingue : { - les types de sets ordonnés - non ordonnés,
  - les types de sets obligatoires - optionnels,

  . Un type de set est dit ordonné si, dans chacune de ses occurrences, les occurrences du record "membre" sont ordonnées suivant un des types de champs de ce record ;
  Le type de set sera réputé non ordonné dans le cas contraire
  . Un type de set est obligatoire si toute occurrence d'un record "membre" de ce type de set doit obligatoirement appartenir à une occurrence de ce set ;
  Dans le cas contraire, le type de set sera optionnel.

représentation graphique :



Le sous-schéma logique : (S.S.L.):

- C'est un ensemble de records et de sets constituant les données permanentes consultées ou mises à jour par une tâche.
  Rmq. 1 : 1 S.S.L. peut être commun à plusieurs tâches.
  Rmq. 2 : un record ou un set peut appartenir à plusieurs S.S.L.

L''area' logique :

-C'est un ensemble d'occurrences de records et de sets rattachées à un même site et pouvant constituer une unité de blocage de ressources (c-à-d. que si un record ou un set est susceptible d'être modifié par une tâche, l'"aréa" correspondante sera bloquée) pendant tout le déroulement de la tâche.

Nous verrons dans le chapitre suivant, les règles de passage du niveau conceptuel au niveau logique.

M.L.T.:modèle logique des traitements. (2)

Rappel:

le modèle logique des traitements (M.L.T.) précise la répartition des traitements entre homme(s) et machine(s) ; pour les traitements machine, il précise également la répartition entre temps réel et temps différé.

La structure logique des traitements est fort hiérarchisée ; c'est ainsi que l'on a la décomposition :

procédure fonctionnelle—phase—tâche—traitement.

Ces 4 concepts sont fortement liés. Nous les présenterons donc ensemble, en y ajoutant le concept fondamental de poste.

Procédure fonctionnelle ou procédure :

C'est un ensemble de traitements (1) concourant à l'élabora-

---

(1) Cfr. ci-après.

(2) Rappel : Les définitions des concepts des différents modèles de MERISE s'inspirent du 'Glossaire pour les données et les traitements'. H. TARDIEU et H. HECKENROTH.

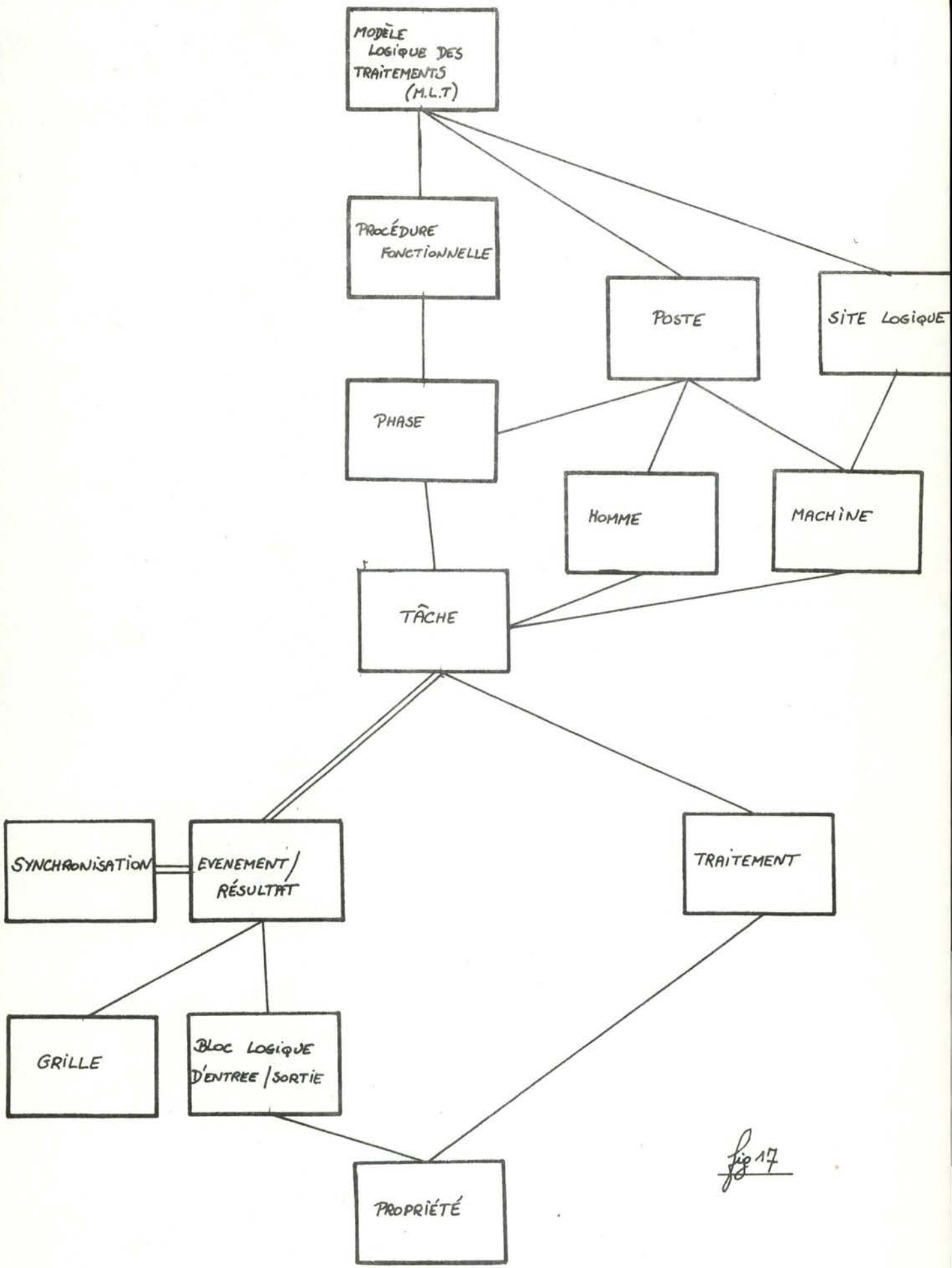

MODÈLE LOGIQUE DES TRAITEMENTS (M.L.T)
PROCÉDURE FONCTIONNELLE
POSTE
SITE LOGIQUE
PHASE
HOMME
MACHINE
TÂCHE
SYNCHRONISATION
EVENEMENT / RÉSULTAT
TRAITEMENT
GRILLE
BLOC LOGIQUE D'ENTREE / SORTIE
PROPRIÉTÉ

fig 17

tion d'un ou plusieurs résultats naturels (1) en réponse à la sollicitation d'une classe d'événements naturels. (1)

- La procédure peut se décomposer en phases et en tâches (1).

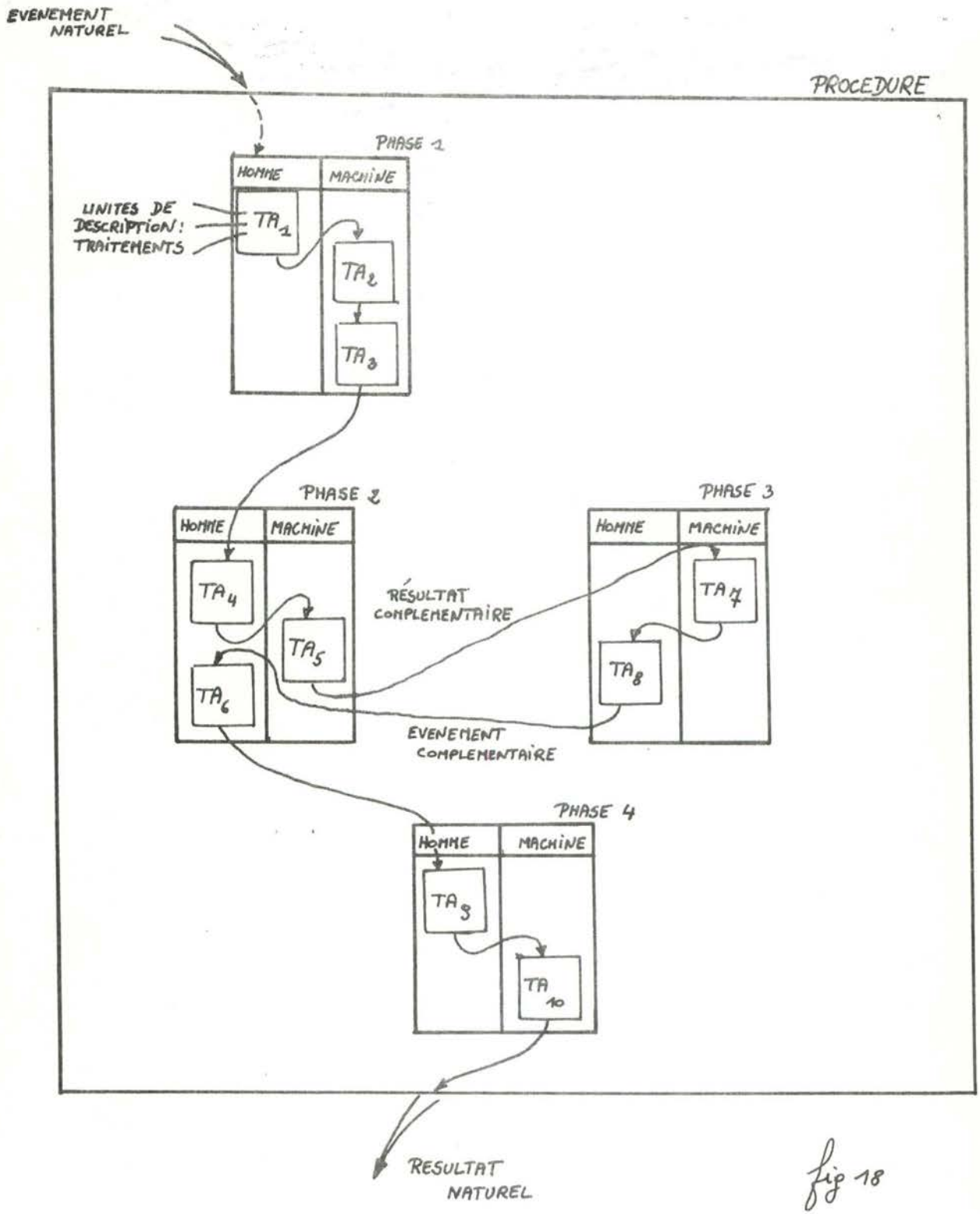


fig 18

(1) Cfr. ci-après.

Evénement (résultat) naturel : événement (résultat) caractéristique d'un domaine d'activité (1) ou d'une grande fonction (1) , du fait des lois et des usages ou des fins propres de l'organisme.

Ex. : commande client, rupture de stock , fin de mois...

domaine d'activité : subdivision logique de l'organisme regroupant un ensemble d'activités complémentaires concourant à la réalisation complète d'une classe de biens ou de services ou à la prise en charge d'une population déterminée.

Ex. : pour une société d'assurances : auto; vie; incendies et risques divers...

grande fonction : subdivision logique de l'organisme regroupant un ensemble d'activités homogènes concourant à un même but :

Ex. : fonctions administrative, commerciale, de production..

EX. DE PROCÉDURES :

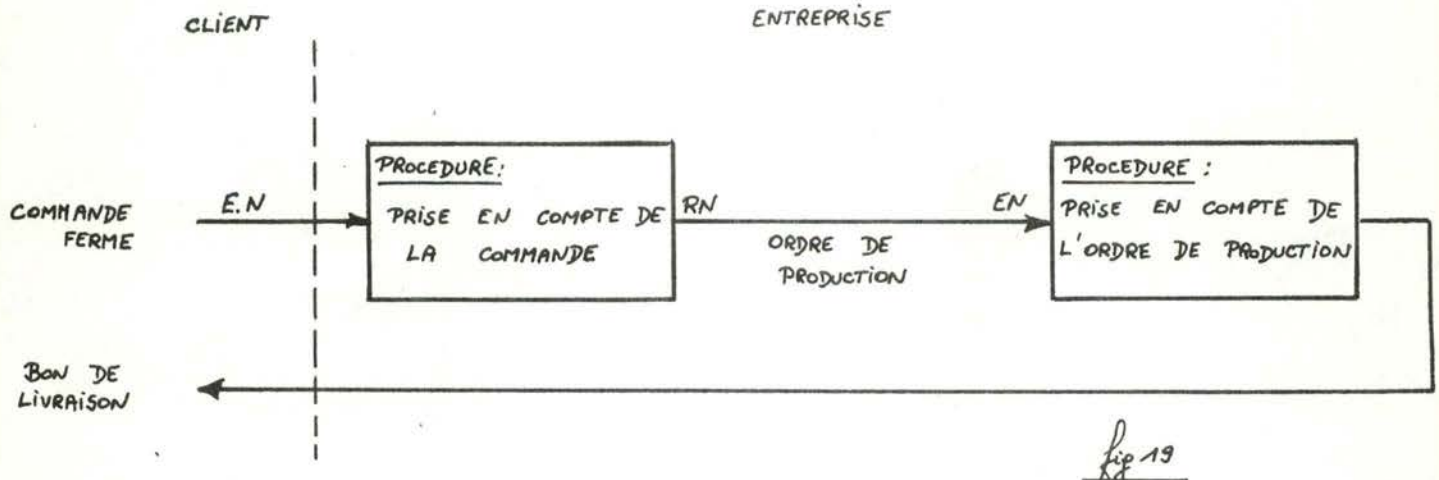


fig 19

Poste :

- Pour des raisons d'organisation, les traitements composant

(1) Cfr. ci-après.

une procédure, sont en général répartis en plusieurs postes.

- Centre d'activité élémentaire de l'organisme, le poste comprend tout ce qui est nécessaire (hommes, machines, espace, outillage...) à l'exécution de tâches (1) définies, dont certaines seront automatisées et d'autres non.
- En général, un poste, à un instant donné, comprend un homme et des moyens de traitement de l'information. Il peut être composé de plusieurs hommes se partageant une même machine ou encore, à la limite, aucune machine (tâches totalement manuelles) ou aucun homme (tâches totalement automatisées).
- Comme il est indiqué sur la fig. 18 p. 59.

  - à une phase correspond un poste.
  - à une tâche correspond un homme ou une machine.

Phase :

- C'est un ensemble logique de traitements (1) exécutés consécutivement par un poste dans le cadre d'une procédure sans qu'il soit possible pour ce poste de prendre en compte (sauf interruption) une nouvelle occurrence du type d'événement qui a initialisé cette séquence avant la fin de celle-ci, ni une occurrence d'un autre type d'événement qui initialiserait une autre séquence de même nature.

  Rmq 1 : possibilité de déroulement parallèle de phases différentes.

  Rmq 2 : lors du déroulement d'une phase, le poste peut faire appel à des interventions externes à ce poste, c-à-d. émettre des résultats intermédiaires vers d'autres postes et en recevoir des événements complémentaires (Cfr. p. 59 fig. 18).

Tâche :

- Sous-ensemble logique de la phase, elle est le résultat de l'affectation des traitements (1) à l'homme ou à la machine.

---

(1) Cfr. ci-après.

- Elle est l'ensemble des traitements (1) successifs affectés à l'un ou à l'autre et dont le déroulement ne nécessite ni l'intervention dans le temps de l'un sur l'autre, ni l'attente du déblocage de certaines ressources.
- On confère donc à la tâche un caractère d'unité dans le temps.
- Elle dispose de toutes les ressources nécessaires à son exécution, notamment RECORDS et SETS du M.L.D.
- La tâche est activée par un événement et délivre un ou plusieurs résultats.
- Chaque occurrence d'événement traitée ne déclenche pas forcément l'exécution des mêmes traitements.

Traitements :

- C'est une unité de description à un niveau logique de l'action exercée sur les données par le système;
  Cette description des traitements est orientée le plus souvent vers la programmation du S.I.
- Les traitements traduisent en fait les règles de gestion du M.C.T. en envisageant notamment toutes les variantes d'événements.
  Ils se basent et utilisent un certain nombre de primitives : actions élémentaires sur les données que l'on détaillera au chapitre IV
  Ex. de primitives :
  - accès à un record connaissant la valeur de sa clé,
  - ajout ou suppression d'une occurrence d'un record,
  - accès maître →1er membre d'un set,
  - parcours d'une occurrence de set,
  - ...

A ces cinq concepts s'ajoutent encore ceux de :

Synchronisation, événements, résultats.

Mêmes concepts que ceux développés pour le M.C.D.

Bloc logique (d'entrée/sortie) :

- Subdivision logique d'un événement ou d'un résultat, le bloc logique (BLE ou BLS) est constitué d' un ensemble de propriétés.
- Par construction, ces propriétés sont saisies simultanément.
- Un BLS peut évidemment devenir à son tour un BLE (cas d'un résultat qui devient un événement par la suite) : la réciproque peut être vraie si le traitement subi par le BLE se limite à un contôle.

Grille :

- C'est une description détaillée du rapport et du graphisme d' un ou plusieurs événements et/ou résultats.
- La description est un ensemble organisé de constantes; elle peut contenir les règles d'assemblage de ces constantes, avec les différents blocs logiques.
  La description permet l'édition de dessins d'état, d'écrans...

Site :

- C'est un ensemble de moyens de traitement automatisé établis en un lieu donné (moyens éventuellement mobiles)

Exemple de M.L.D. MERISE.

Reprenons le cas "PETITPAS" et plus particulièrement le M.C.D. construit ci-avant (cfr. p.22) et élaborons le M.L.D. Cette transformation du M.C.D. en M.L.D. est largement facilitée par l'utilisation d'un ensemble de règles que nous exposerons au chapitre III 2. p.92 (MAPPING M.E.D., M.C.D. ⟶ S.S.L., M.L.D.)

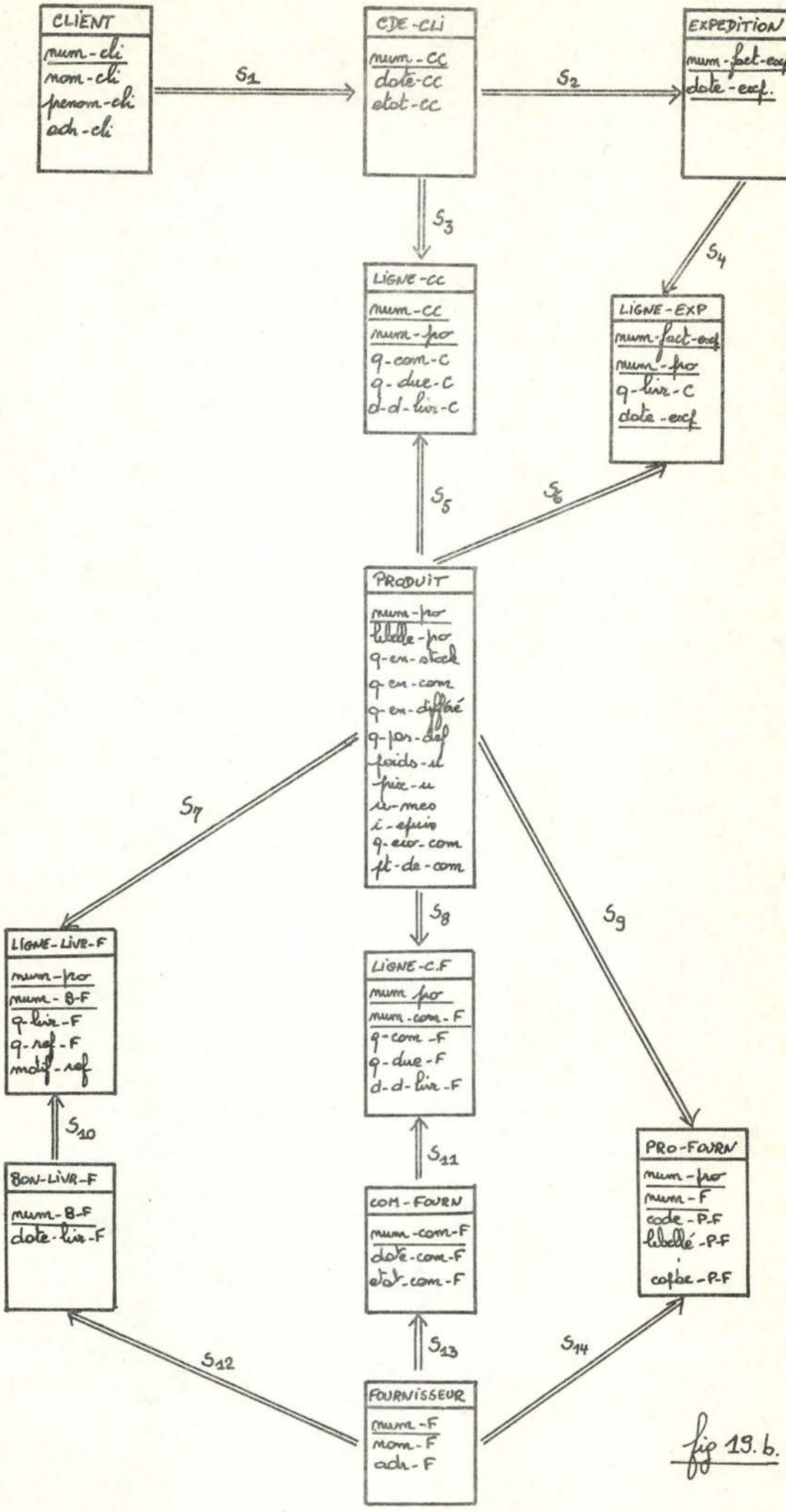
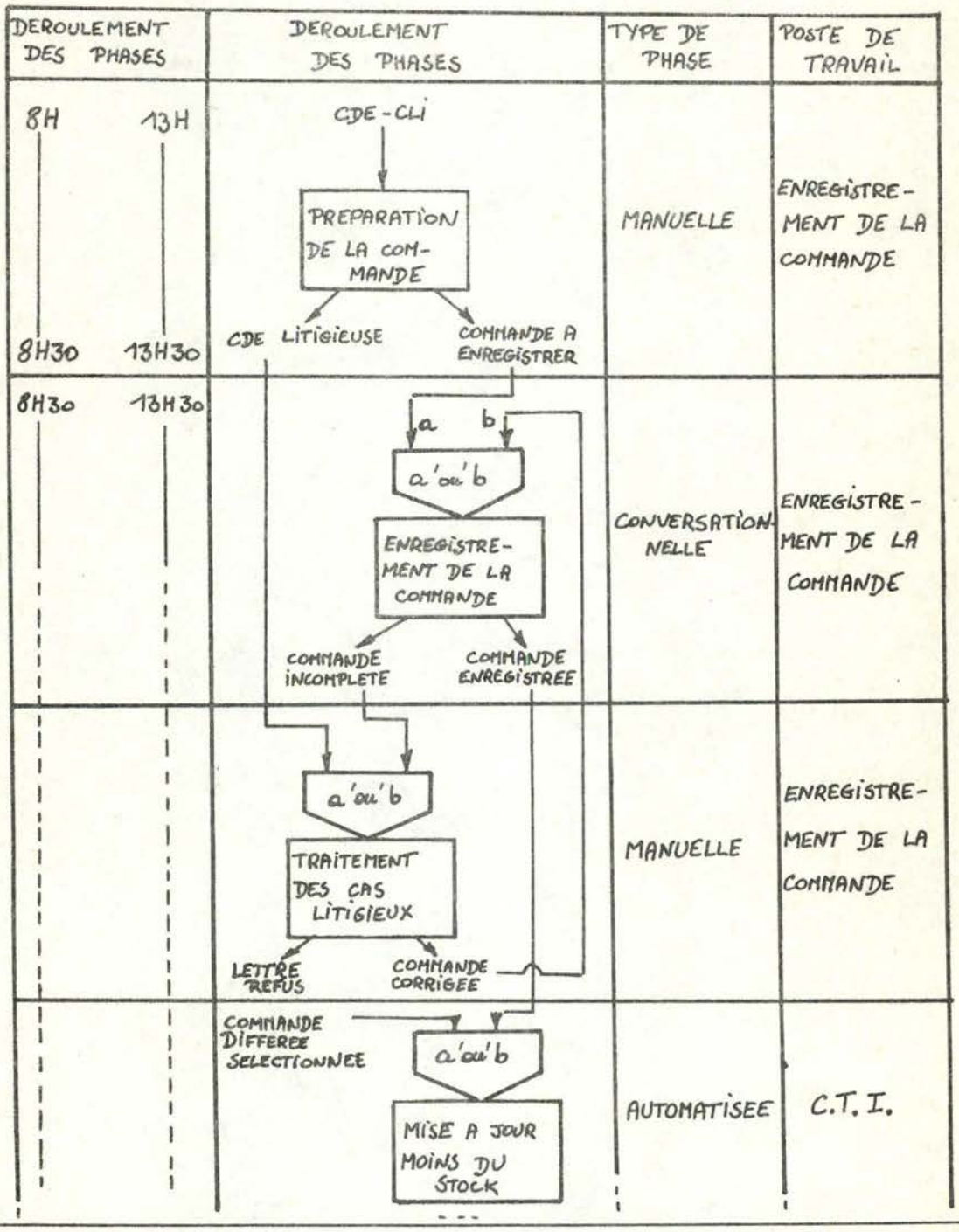

CLIENT
num-cli
nom-cli
prenom-cli
adr-cli
S1
CDE-CLI
num-CC
date-CC
etat-CC
S2
EXPEDITION
num-fact-exp
date-exp.
S3
S4
LIGNE-CC
num-CC
num-pro
q-com-C
q-due-C
d-d-liv-C
LIGNE-EXP
num-fact-exp
num-pro
q-liv-C
date-exp
S5
S6
PRODUIT
num-pro
libelle-pro
q-en-stock
q-en-com
q-en-différé
q-por-def
poids-u
prix-u
u-mes
i-epuis
q-exp-com
pt-de-com
S7
S8
S9
LIGNE-LIVR-F
num-pro
num-B-F
q-liv-F
q-ref-F
motif-ref
LIGNE-C.F
num pro
num-com-F
q-com-F
q-due-F
d-d-liv-F
PRO-FOURN
num-pro
num-F
code-P.F
libellé-P.F
cofbe-P.F
S10
S11
BON-LIVR-F
num-B-F
date-liv-F
COM-FOURN
num-com-F
date-com-F
etat-com-F
S12
S13
S14
FOURNISSEUR
num-F
nom-F
adr-F

fig 19.b.

Exemple de M.L.T. MERISE.

Il nous faut donc maintenant affiner la description des traitements faite au niveau du M.C.T. et préciser quelles sont les différentes procédures, phases et tâches relatives au cas envisagé.

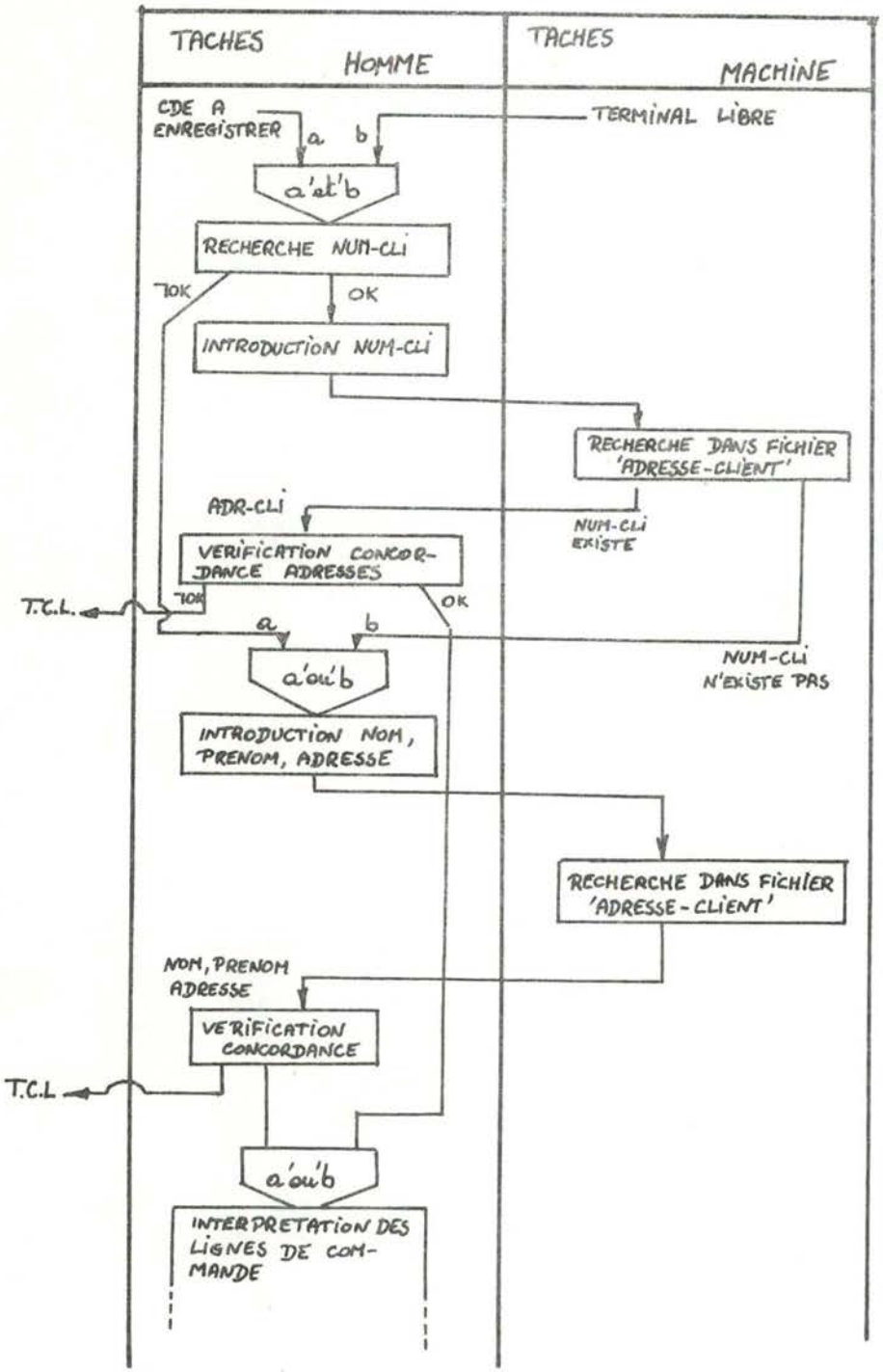
ex : Procédure : Traitement-des cdes-clients :



(1) Les opérations du M.C.T. ont été ici assimilées aux phases.

De plus, on détaille chacune des phases de la façon suivante :

ex : phase : enregistrement de la commande.



T.C.L: TRAITEMENT DES CAS LITIGIEUX

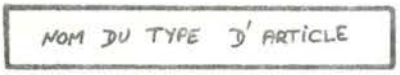
Il resterait donc ici à donner -le dessin du bon-de-commande
-les dessins des écrans.
(ce que MERISE appelle les grilles).

II 2.2. 'NAMUR' : Modèles de niveau logique.

Modèle des accès logiques (1) :

Article et type d'article :

- L'article est une entité d'information enregistrée qui peut faire l'objet d'une demande d'accès de la part d'un programme.
- Les articles d'une B.D. sont discernables et ce, indépendamment des valeurs d'items qui leur sont associées.
- Tout article appartient à un et un seul type qui en définit les propriétés générales.
- Représentation graphique d'un type d'article :

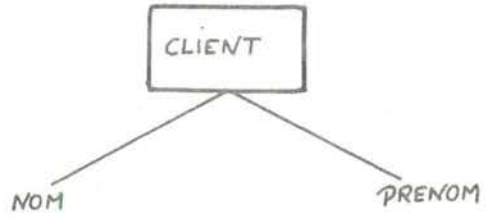


Valeur d'item et item :

- La valeur d'item est une information qui désigne l'état d'une variable telle qu'une propriété d'un système ou d'un élément d'un système.
  Rmq. : il importe de ne pas oublier deux valeurs singulières d'un item : "absent" et "inconnu".
- L'item, quant à lui, est un type d'information défini par un ensemble de valeurs. Il est associé à au moins un type d'article.
  Représentation graphique :

  Pour un item, on aura : NOM-DE-L'-ITEM-EN-LETTRES-CAPITALES
  - L'association d'un item à un type d'article sera représentée par un arc reliant les représentations des deux types.

EX:

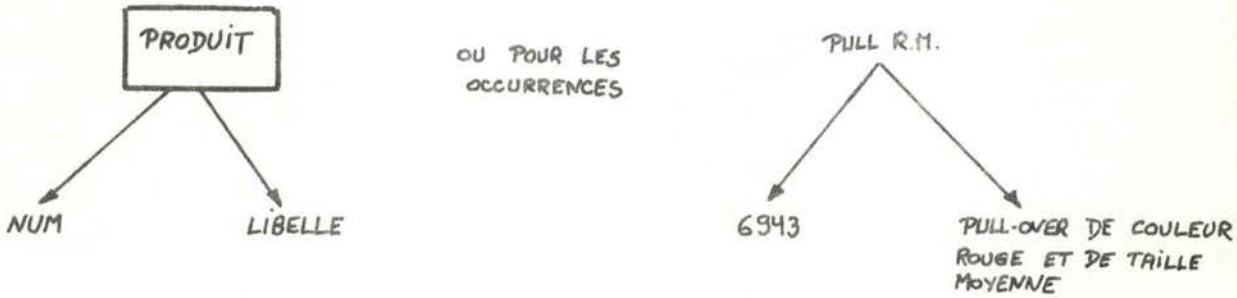


Généralement on admet qu'il existe toujours un mécanisme

(1) J.L. HAINAUT. "Un modèle de description de fichiers : le modèle d'accès."

d'accès du type d'article vers l'item associé; il est toutefois préférable de le matérialiser en orientant l'arc :

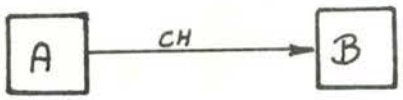
EX:



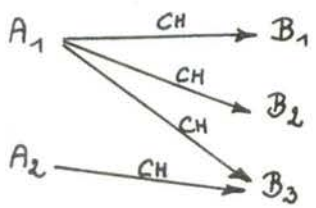
Chemin et type de chemin d'accès inter-articles:

- Le chemin d'accès est un mécanisme qui associe un article dit "origine" à n (n> 0) articles dits "cibles". Ainsi, à partir de l'article origine,on doit être en mesure d'accéder successivement aux différents articles cibles associés.
- Tout chemin appartient à un et un seul type qui en définit les propriétés générales.
- Un type de chemin, caractérisé par les types d'articles qu'il associe, porte un nom (l'absence de nom sera considérée comme un "nom vide").
- Il ne peut exister deux types de chemins portant le même nom (ou absence de nom) entre un type d'article (origine) et un autre (cible).
- Les représentations graphiques se feront de la manière suivante :

EX:



POUR LES TYPES.



POUR LES OCCURRENCES.

Fichiers et B.D. :

- Un fichier est une collection dynamique d'articles; on lui associe un ou plusieurs types d'articles. Un type d'article peut appartenir à un ou plusieurs fichiers; un article ne peut,

lui, appartenir qu'à un et un seul fichier.

- Une B.D. est la collection de tous les aticles qui, à un instant déterminé, décrivent complètement un système réel. Elle peut contenir un ou plusieurs fichiers mais un fichier ne peut appartenir qu'à une et une seule B.D.
- Toute B.D. contient un et un seul article d'un type déterminé : le SYSTEME.
- C'est un point d'entrée privilégié dans la B.D.; il peut être origine de chemins d'accès mais en aucun cas cible d'un chemin d'accès.

A ces concepts de base, on peut en ajouter d'autres:

Clé d'accès :

- Item ou liste d'items d'un type d'article tel qu'il existe un mécanisme qui permette d'accéder aux différents articles auxquels est associée une valeur déterminée de la clé.
- On la représentera comme suit :



Item(s) identifiant(s) :

- C'est un item ou une liste d'items d'un type d'article tel qu'il n'existe pas plus d'un article qui soit associé à une même valeur de cet item ou de cette liste d'items.
- Représentation graphique :



Connectivité de l'association d'un item à un type d'article :

Les connectivités retenues, exprimées dans le sens article-item, sont les suivantes :

M-N (many-to-many) : à un article peuvent être associées plusieurs valeurs d'item et une valeur d'item peut être associée à plusieurs articles.



Dans ce cas, l'item est répétitif et non identifiant.

1-N (one-to-many) : à un article peuvent être associées plusieurs valeurs d'item mais une valeur d'item ne peut être associée qu'à un seul article.



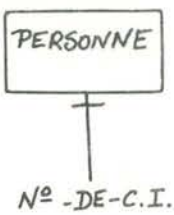
Cas d'un item répétitif et identifiant.

N-1 (many-to-one) : un article ne peut posséder qu'une valeur de l'item mais une valeur d'item peut être associée à plusieurs articles.



Cas d'un item simple et non identifiant.

1-1 (one-to-one) : un article ne peut posséder qu'une valeur de l'item et une valeur d'item ne peut être associée à plus d'un article.

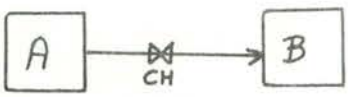


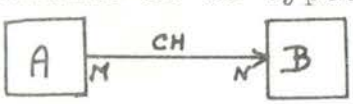
Cas d'un item simple et identifiant.

Connectivité d'un type de chemins :

On retiendra :

- les types de chemins M-N : un chemin peut contenir plus d'un article cible et un article peut être cible dans plusieurs chemins de ce type.





- les types de chemins 1-N : un chemin peut contenir plusieurs articles cibles mais un article ne peut être cible que d'un seul chemin de ce type.



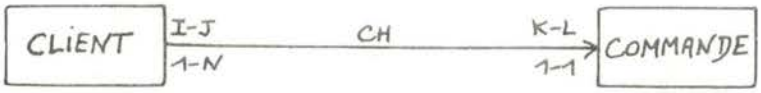
- les types de chemins N-1 : un chemin ne peut contenir qu'un seul article cible mais un article peut être cible de plusieurs chemins de ce type.



- les types de chemins 1-1 : un chemin ne peut contenir qu'un article cible et un article ne peut être cible que d'un chemin de ce type.



Rmq. : la connectivité, ainsi que certaines contraintes d'existence (cfr. ci-après : type de chemin fort ou faible) peuvent être exprimées par un quadruplet d'entiers I,J,K,L décrivant la cardinalité d'un type de chemin : I,J donnent les nombres minimum et maximum d'articles cibles qui peuvent être associés à un article origine et inversément pour K,L.



Type de chemin fort ou faible :

Un type de chemin est dit fort pour l'un de ses types d'articles, cibles ou origines si tout article de ce type doit,à tout moment, être cible ou origine d'au moins un chemin non vide de ce type; il est déclaré faible pour les types d'articles ne subissant pas cette contrainte.
On représentera cette contrainte de la façon suivante :



un trait barre le type de chemin d'accès à proximité du type d'article pour lequel ce type de chemin est fort.

Mode d'appartenance à un chemin d'accès :

Des contraintes peuvent s'appliquer lors de l'insertion ou du retrait d'un article dans un chemin.
Lors de l'insertion d'un article, on distinguera l'appartenance

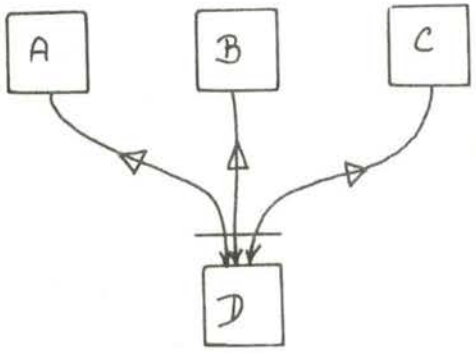
- automatique : l'insertion dans un chemin est alors automatique,
- manuelle : l'insertion dans un chemin est laissée au soin du programmeur.

Lors du retrait d'un article, on distinguera l'appartenance :

- fixe : un article inséré ne pourra plus être retiré du chemin,
- obligatoire : un article inséré ne pourra être que transféré d'un chemin à un autre chemin du même type,
- facultative : l'article peut à tout moment être détaché du chemin et éventuellement être ré-inséré dans un autre chemin du même type.

Groupe de types de chemins identifiant :

Si l'intersection des cibles de n chemins ($n \geqslant 2$) ne contient pas plus d'un article, on pourra dire que, pour ce type d'article, les types de chemins envisagés sont "identifiant".
Représentation graphique :



Tels sont les principaux concepts du modèle des accès logiques. Signalons, pour terminer, qu'il est possible de généraliser la notion de chemin d'accès.

En effet, nous avons vu - l'accès aux valeurs d'un item à partir d'un article,
- l'accès aux articles associés à une même valeur de clé,
- l'accès aux articles cibles d'un chemin d'accès,

la généralisation consistera à admettre les chemins d'accès :

- d'article vers article,
- d'article vers valeur d'item,
- de valeur d'item vers article,
- de valeur d'item vers valeur d'item.

Exemple de schéma des accès logiques 'NAMUR'. (1)

En reprenant le S.C.D. donné par ailleurs (p.35) et en lui appliquant des règles de transformation que nous étudierons au chapitre III 2. p.96, nous obtenons assez facilement le schéma des accès logiques possibles qui utilise les concepts que nous venons de voir.

(1) cfr. J.L. HAINAUT : "Analyse du cas 'PETITPAS' - Dossier d'analyse organique - Premiére partie : la base de données.".

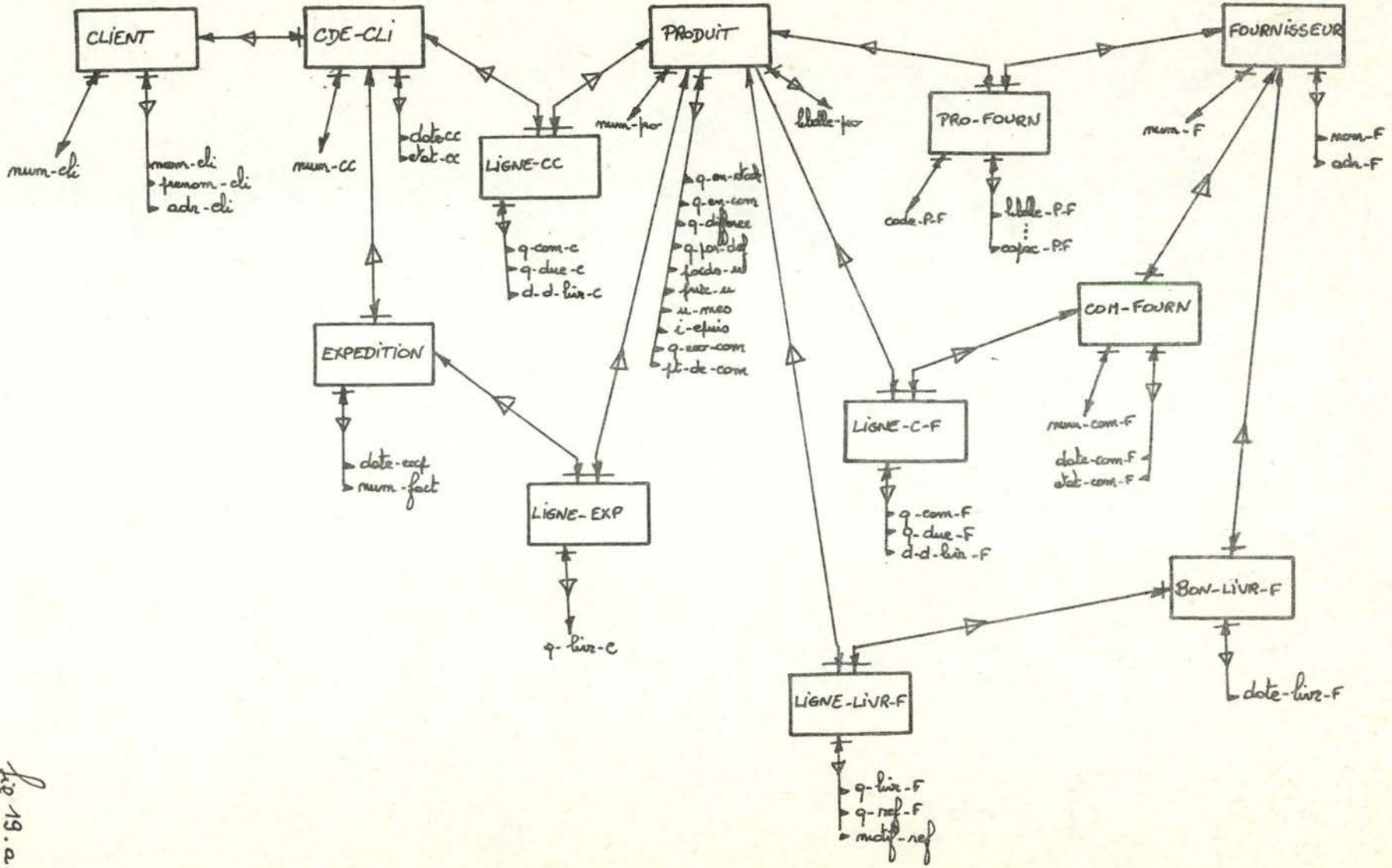

CLIENT
num-cli
nom-cli
prenom-cli
adr-cli
CDE-CLI
num-cc
date-cc
etat-cc
LIGNE-CC
q-com-c
q-due-c
d-d-liv-c
PRODUIT
num-pro
libelle-pro
q-en-stock
q-en-com
q-disponible
q-pour-def
seuil-alerte
unite-vente
prix-unitaire
q-reservee
pt-de-com
PRO-FOURN
code-P.F
libelle-P.F
coefac-P.F
FOURNISSEUR
num-F
nom-F
adr-F
EXPEDITION
date-exp
num-fact
LIGNE-EXP
q-liv-c
COM-FOURN
num-com-F
date-com-F
etat-com-F
LIGNE-C-F
q-com-F
q-due-F
d-d-liv-F
BON-LIVR-F
date-livr-F
LIGNE-LIVR-F
q-liv-F
q-ref-F
motif-ref

fig 19.a.

Modèle logique des traitements.

Le niveau logique des traitements conserve la découpe en applications-phases-fonctions que nous avons effectuée au niveau conceptuel.
Il apporte cependant un certain nombre de transformations et de précisions à ce qui a déjà été vu.

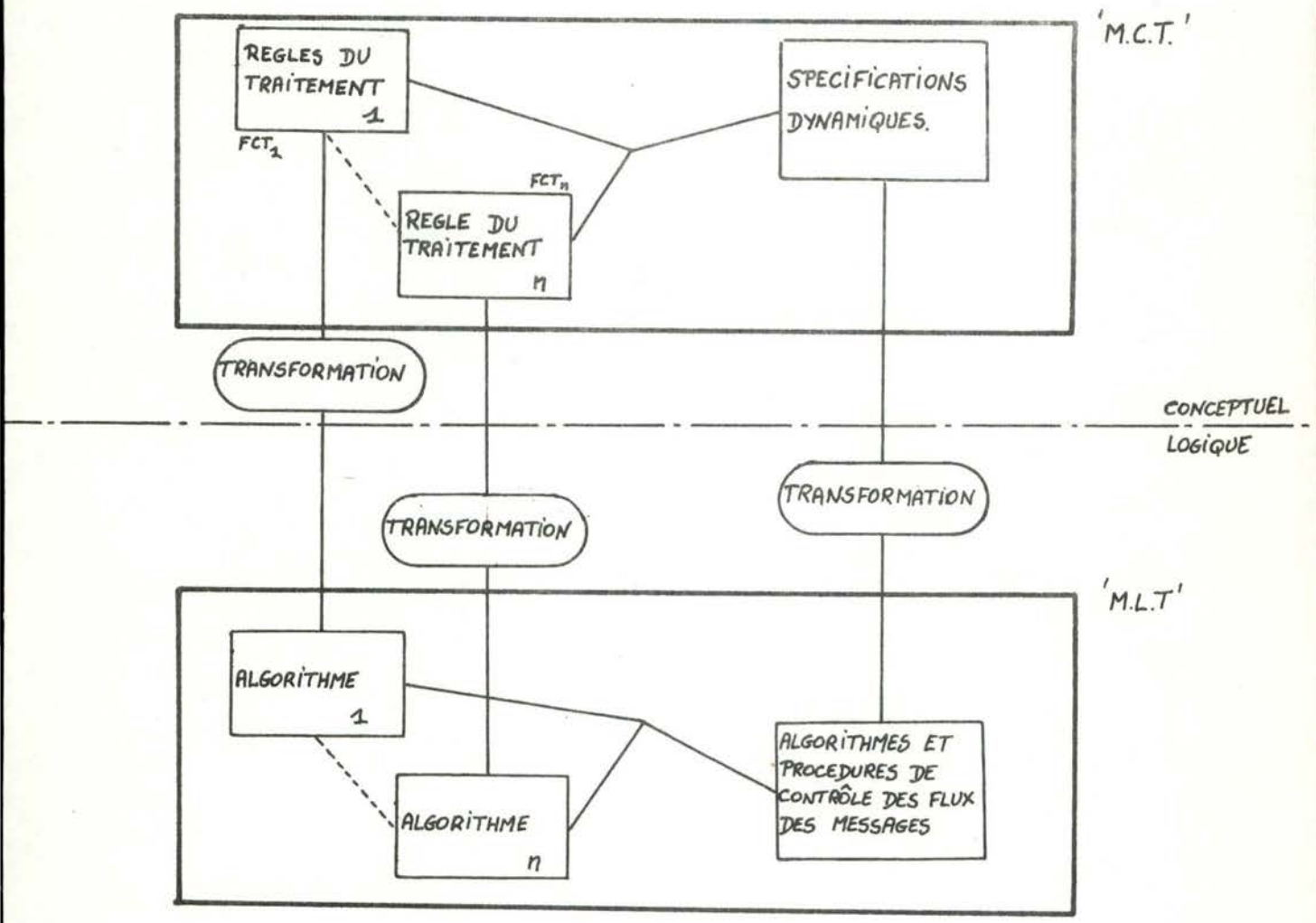


fig 19. b

C'est ainsi que le 'M.L.T.' comprend :

- un ensemble d'algorithmes de traitement, transformation des règles contenues dans les processus (uniquement de type 'fonction') du niveau conceptuel,

- un ensemble d'algorithmes et de procedures organisationnelles de contrôle des flux de messages découlant directement des spécifications dynamiques du niveau conceptuel.

- Les algorithmes de traitement utilisent un certain nombre de primitives, opérations de base qui s'exercent sur les différents types de données que l'on a définis et que l'on met à la disposition du programmeur et de l'utilisateur.
A titre d'exemple (1), on dispose des primitives :

  - d'accès au premier(dernier) article d'un chemin,
  - d'accès à l'article cible suivant (précédent) dans un chemin,
  - d'accès direct dans un chemin,
  - d'accès à des articles par clés d'accès,
  - d'accès aux valeurs d'items d'un article,
  - de création et de suppression d'un article,
  - de modification de valeurs d'items,
  - d'insertion ou de retrait d'un article dans un chemin
  - de transfert d'un chemin à un autre.

Il est donc facile,connaissant les règles de traitement (niveau conceptuel) et les primitives, de rédiger l'algorithme qui parcourra une structure de données et y apportera les modifications voulues.

- Du point de vue organisationnel, c'est aussi au niveau logique des traitements que l'on établira l'architecture des moyens de réalisation et que l'on décrira les moyens disponibles utilisables par les processus lors de leur exécution.
On définit à ce propos (2) :

  Le processeur : objet capable d'exécuter les processus. On peut considérer comme processeurs, un employé, une machine, un service de comptabilité....
  Rmq. : un processeur peut requérir, lors de l'exécution d'un processus, l'utilisation d'autres processeurs; ceux-ci deviendront donc "processeurs auxiliaires" du premier.

---

(1) pour plus de précisions : cfr. chap.IV.2.1.8.

(2) Cfr."D.S.L. Users'Manual" F.BODART et Y.PIGNEUR.

Les ressources : objets qui peuvent être requis par un processeur pour exécuter un processus.
On peut éventuellement distinguer :

- les ressources ré-utilisables : ressources qui retournent à la disposition du système dès que le processus est terminé.
  Ex. : les terminaux, un outil, un employé.
  Rmq. : il n'y a pas de différence de nature entre un processeur et une ressource ré-utilisable mais seulement une différence de degré d'analyse : le premier peut exécuter un processus et utiliser des ressources ré-utilisables et consommables (c'est donc une ressource ACTIVE) ; la seconde ne le peut pas

- les ressources consommables : ressources qui sont irrémédiablement consommées par les processeurs lorsqu'ils exécutent un processus.
  Ex. : le papier, l'énergie....

Ainsi, on peut dresser le modèle des ressources suivant :

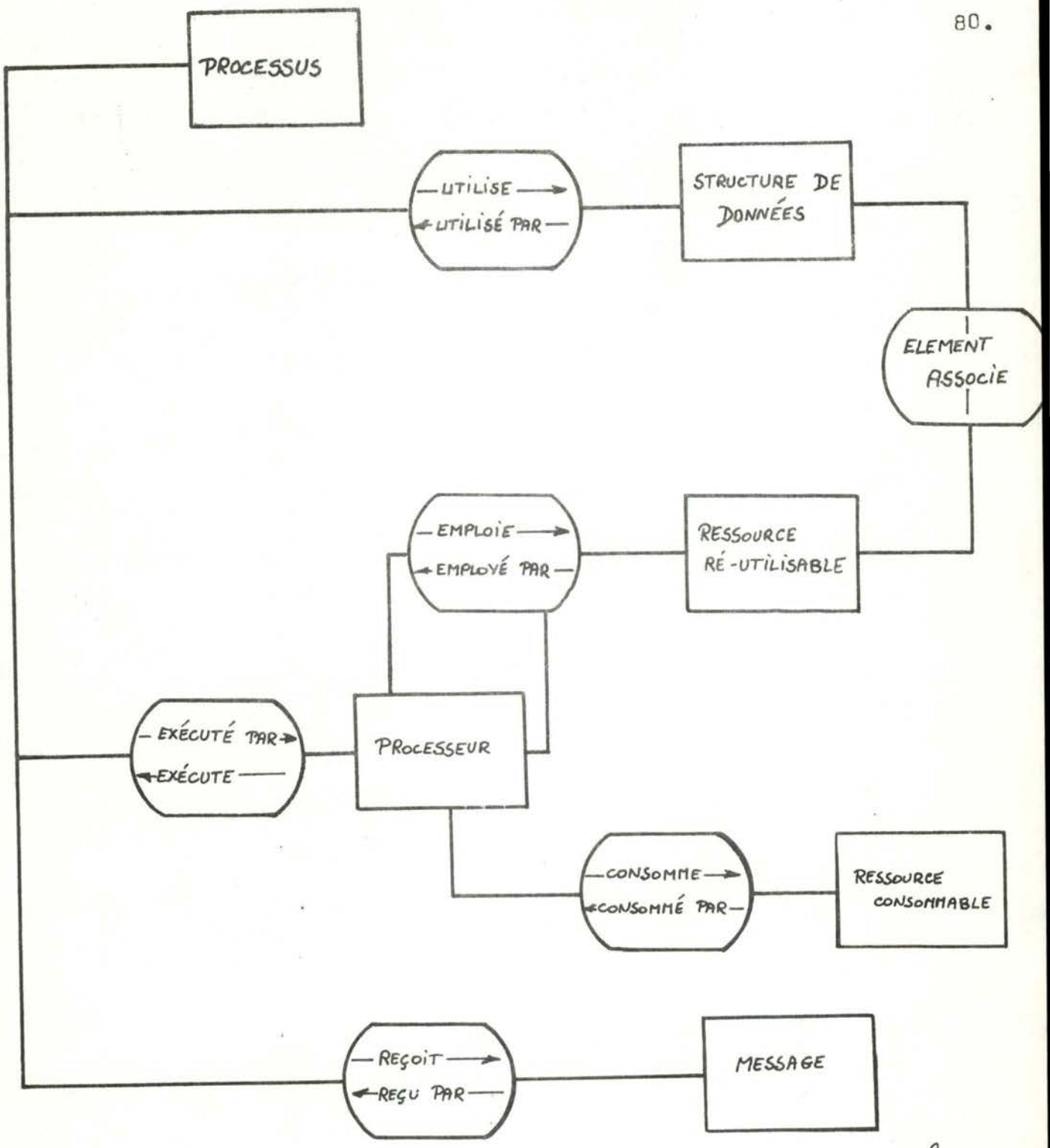


fig 20

On peut complexifier ce modèle en définissant notamment :

- des disponibilités de ressources,
- des calendriers d'activation,
- des priorités (pour les processeurs),
- des nombres de points d'entrée (physiques ou logique (pour les processeurs et les ressources ré-utilisables)),
- des relations de consommation, ...

Exemple de modèle logique des traitements : 'NAMUR'.

Il reste donc à préciser la modélisation des traitements que l'on a pu effectuer au niveau conceptuel (cfr. p.48-49-50).

Nous ne détaillerons cependant pas ici les algorithmes de traitement. Ce point sera développé dans la suite (cfr. p.127 et suivantes.) et un exemple sera donné à cette occasion.

Par contre, les spécifications des ressources utilisées par les différents processus que l'on a identifiés, doivent être faites au niveau logique.

Ainsi, pour le cas 'PETITPAS', nous avons par exemple :

1. Fonction : vérification-signature
    2. Ressource rr-service-enregistrement-cdes (1) requise à un taux d' '1';
        3. Ressource rr-employé requise à un taux d' '1' par le service-enregistrement-cdes;
        3. Ressource rr-manager requise au taux de 'sp-1';(2)
        3. Ressource rr-terminal-service-enreg requise au taux de 'sp-2';
            4. Ressource rr-cpu-mini-computer requise au taux de 'sp-3';

ou encore :

1. Fonction : vérification-corps-cdes
    2. Ressource rr-service-enregistrement cdes requise au taux d' '1' par la fonction 'vérification-corps-cdes;
        3. Ressource rr-employé requise au taux d' '1' par le service-enregistrement-cdes;
        3. Ressource rr-manager requise au taux de 'sp-4' par rr-service-enregistrement-cdes pour exécuter le processus 'maj-de-l'ident-client' requise au taux de 'sp5' par rr-service-enregistrement-cdes ;

---

(1) on n'a pas fait ici la différence entre 'ressource-reutilisable (rr) et 'processeur' (pr.).

(2) sp = paramètre à définir ailleurs.

pour exécuter le processus 'contrôle-des-lignes-cdes'

3. Ressource rr-terminal-service-enreg
requise..... par rr-service-enregistrement-cdes;

4. Ressource rr-cpu-mini-computer
requise..... par rr-terminal-service-enreg....;

ou encore :

1. Fonction : maj-moins-stock

2. Ressource rr-cpu-mini-computer
requise..... par le processus 'maj-moins-stock';

2. Ressource rr-fichier-produits
requise au taux de '1' par le processus 'maj-moins-stock';

2. Ressource rr-fichier-cdes-différées
requise au taux de '1' par le processus 'maj-moins-stock';

les ressources pouvant être définies, par exemple, comme suit :

Ressource-rr-service-enregistrement-cdes;
Capacité : 'sp-10';
Partageable parmi 'sp-11';
Disponible suivant 'calendrier-1';
Requiert 1 rr-employé;
Requiert 'sp-12' rr-manager pour exécuter les processus :
- 'maj-ident-client',
- 'contrôle-des-lignes-cdes';
Unité de mesure : 'sp-13';

II 2 3 Commentaires .

Pour ce qui est des modèles propres aux deux approches, on constate de nouveau quelques petites différences mais on ne peut pas affirmer que l'on est en présence de deux positions radicalement opposées !

Pour les données :

On remarque tout d'abord une similitude de concepts . C'est ainsi que l'on a :

- le record	- l'article
- le champ	- la valeur d'item
- le set	- le chemin d'accès inter-articles
- le set ordonné ou non ordonné	- ordre ou non dans les articles cibles d'un chemin
- le set optionnel ou obligatoire	- le chemin faible ou fort pour un type d'article cible
- l'aréa logique	- le fichier ou la B.D.

et ce, respectivement pour MERISE et 'NAMUR' .

Les concepts de MERISE se limitent, à peu de choses près, à ceux que je viens d'énumérer .

On est donc en présence d'un M.L.D. très simple, inspiré d'ailleurs de CODASYL . En fait, lorsque l'on prendra en compte l'activité sur les différents S.S.L (et donc sur le M.L.D.), on introduira encore de nouvelles considérations telles que :

- clé d'accès à un record,
- connectivités des sets,
- cardinalités des records, ... (cfr. chap IV)

si bien que l'on obtient un M.L.D. qui ressemble très fort au schéma des accès logiques proposé par 'NAMUR' .

Il n'y a donc pas encore, à ce niveau, de différences fondamentales entre les deux approches .

Pour les traitements :

a) MERISE fait, au niveau du M.L.T., la décomposition des opérations en :
- procédure fonctionnelle,
- phase,
- tâche,
- traitement,

et ce, sur base d'un certain nombre de définitions parfois peu constructives et qui ne permettent pas toujours de faire la discrimination dans les opérations .
'NAMUR', quant à elle, proposait déjà la décomposition des processus dès le niveau conceptuel (cfr. nomenclature des traitements p. 40) .

b) Les deux approches proposent l'élaboration d'algorithmes qui se basent sur l'utilisation de primitives données . Cette élaboration se fera à l'aide des règles de gestion (MERISE) ou de traitement ('NAMUR') et décrira les actions effectuées sur une certaine structure de données . Nous aurons l'occasion de revenir sur ce point (cfr. chap IV) .

c) Les deux approches permettent également la description des choix organisationnels, à savoir l'affectation des traitements aux différents moyens logiques de réalisation : hommes, machines, postes - processeurs, ressources .
Rmq. : Si MERISE est assez explicite quant à la spécification des processeurs alloués aux différents traitements, elle ne donne aucun détail sur les ressources utilisées lors de l'exécution de ces traitements (ressources consommables ou ré-utilisables).

II 3 . Modèles de niveau physique (p.m.) .

Ces modèles, qui doivent permettre l'élaboration proprement dite de la B.D. et des différents programmes qui l'utiliseront, ne sont pas pris en compte dans le cadre de ce mémoire . Ils sont en effet trop tributaires de considérations purement physiques : type de matériel, S.G.B.D. utilisé, ... et risquerait de nous entraîner trop loin .

Chap. III. Comparaison des correspondances inter-modèles.

Nous allons maintenant essayer de définir les nombreuses correspondances qui existent entre les différents modèles que nous avons exposés au chapitre précédent.
C'est ainsi que nous nous attacherons à décrire :

1) les rapports entre les modèles externes et conceptuels d'un S.I.,
2) les mappings (1) entre les niveaux conceptuel et logique d'un S.I.,
3) les mappings entre les niveaux logique et physique d'un S.I. (p.m.),
4) la cohérence entre les modèles de données et les modèles de traitements.

III.1. Rapports entre les modèles externes et conceptuels d'un S.I.

Ces rapports sont différents suivant l'approche considérée. Il est toutefois bon de rappeler, avant de les étudier, ce qu'est un modèle externe du système. C'est une vue particulière de l'activité de l'organisation qui correspond à un programme, une fonction, une application ... et qui émane souvent d'une certaine classe d'acteurs de l'organisation. Il se caractérise par une structure de données propre (M.E.D.) qui peut être exprimée dans un formalisme particulier (2)(bien que ce ne soit pas le cas dans les approches MERISE et 'NAMUR' qui utilisent respectivement, pour la définition des M.E.D. et des M.C.D., le modèle individuel et le modèle entité-association) ainsi qu'un schéma de traitements (ou d'opérations) qui est le reflet de l'activité d'un sous-ensemble de l'organisation et que l'on pourrait éventuellement désigner par M.E.T.

---

(1) MAPPING : correspondance, processus de passage.
(2) cfr. rapport ANSI-SPARC
"A Gross Architecture For The Next Generation Database Management Systems". G.M. Nijssen.

En ce qui concerne les rapports entre les modèles externes et les modèles conceptuels, nous avons :

Pour MERISE

M.E.D. ⟶ M.C.D.

Comme nous le verrons dans le chapitre suivant relatif à la démarche MERISE, les modèles externes de données doivent être exprimés à l'aide du vocabulaire - c'est-à-dire de la liste des propriétés - du modèle conceptuel brut (M.C.B.).
Ce dernier n'est autre que l'expression structurée des informations du réel perçu communes à tous les acteurs de l'organisation, les règles de structuration étant celles du formalisme individuel (cfr. chap. II).
L'élaboration des M.E.D. est soumise au respect d'un certain nombre de règles :

a) les propriétés externes sont nécessairement des propriétés conceptuelles appartenant au modèle conceptuel,
b) un individu ou une relation externe ne comporte pas de propriétés répétitives,
c) un individu externe peut comporter 0, 1 ou plusieurs identifiants conceptuels,
d) un M.E.D. peut comporter éventuellement des individus, relations et contraintes existant déjà dans le M.C.D.

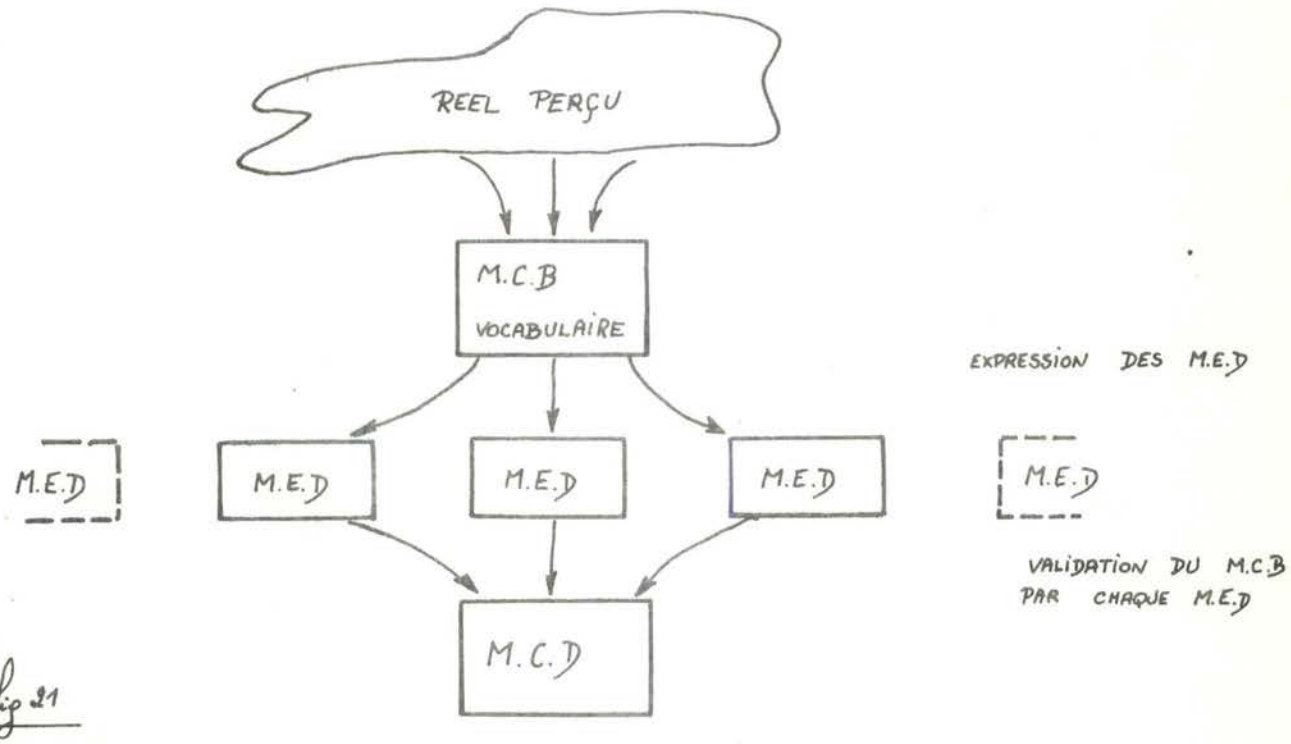


fig 21

Dès que l'on aura élaboré les différents M.E.D., il faudra valider le M.C.B., ce qui aura pour résultat le M.C.D. proprement dit et permettra aussi de prendre en compte le M.E.D. sous la forme équivalente d'un sous-ensemble du modèle conceptuel validé.

La validation consiste à s'assurer que les M.E.D. sont déductibles du M.C.B. en utilisant un jeu de règles que l'on peut appliquer manuellement ou automatiquement. Elle doit être utilisée pendant la phase de conception mais peut être appliquée aussi lorsque le S.I. est déjà opérationnel.

On peut se demander pourquoi il faut valider. Il y a pour cela trois raisons principales :

a) on peut avoir commis des erreurs en passant du réel perçu au M.C.B.,

b) les M.E.D. peuvent avoir introduit de nouveaux individus à partir du M.C.B., ce qui peut avoir pour conséquence l'apparition de nouveaux identifiants dont il faudra tenir compte, notamment pour les fonctions de mise à jour. De plus, des erreurs peuvent être commises au niveau du choix de l'identifiant d'un individu. Un identifiant trop large ou trop fin peut en effet poser quelques problèmes.

ex. : identifiant trop large : - individu : véhicule
- identifiant : modèle

Si on veut faire des mises à jour de véhicules ayant un certain n° d'immatriculation, l'identifiant est insuffisant.

identifiant trop fin : - individu : véhicule
- identifiant : n° d'immatriculation

Si on ne fait que des mises à jour de véhicules d'un certain modèle, l'identifiant 'modèle' aurait suffi.

c) on doit pouvoir faire apparaître des relations sans propriétés, ce qui n'est pas possible à partir du M.C.B. (Il a en effet été construit à partir de propriétés du réel perçu (cfr. chap.IV) et n'a donc que des relations avec propriété(s)).

Si, au cours de la validation, on constate une erreur ou une impossibilité, il faudra :

- soit modifier le M.C.B. et refaire toutes les validations,
- soit modifier le M.E.D. en cause.

Comment effectuer la validation?

Il faut d'abord distinguer - les modèles externes en mise à jour: modèles utilisés par les fonctions qui ont pour but de modifier la B.D.
- les modèles externes en consultation: modèles utilisés par les fonctions qui ne font que consulter la B.D. et ne la modifient jamais.

Validation du modèle conceptuel par les modèles externes en mise à jour.

Elle consiste à examiner le rôle que joue chacune des propriétés invoquées dans un individu ou une relation externe (appartenant à un M.E.).

Une propriété peut - soit identifier une occurrence d'un individu ou d'une relation conceptuel(le) que l'on veut créer ou modifier,
- soit permettre le chargement d'une valeur sur une occurrence d'individu ou de relation conceptuel(le) préalablement identifiée.

Il y a non-validation lorsque :

a) une propriété est destinée à mettre à jour un individu que l'on ne peut identifier. Dans ce cas, 2 solutions :
   - ajouter une propriété externe pour permettre l'identification,
   - supprimer la propriété externe correspondante et renoncer à la mise à jour.

b) une propriété externe n'est destinée à aucun chargement. Il faut donc la supprimer.

c) une propriété externe est un identifiant conceptuel non nécessaire aux mises à jour envisagées. Ici encore, il faut la supprimer.

Validation du modèle conceptuel par les modèles externes en consultation.

Elle repose sur le respect de 3 règles :

a) une propriété externe doit être identique à une propriété conceptuelle. Au pis, elle doit pouvoir être remplacée par

une structure équivalente du modèle conceptuel.

b) une occurrence d'un individu externe, pour être valide, doit être exprimable sans ambiguïté à partir d'occurrences d'individus ou de relations conceptuelles.

c) une relation externe est valide si les individus externes qui composent sa collection sont valides et si elle est construite en respectant les conditions suivantes :
   - elle résulte de la composition de plusieurs relations conceptuelles entre des individus conceptuels identiques aux individus externes cités,
   - elle correspond à une relation conceptuelle dont :
     • la collection est identique ou incluse dans l'ensemble des individus conceptuels invoqués dans les individus externes de la collection de la relation externe,
     • au moins un individu de sa collection apparaît dans chaque ensemble d'individus conceptuels correspondant à chaque individu externe de la collection de la relation externe.

Voici donc quelles sont les relations entre M.E.D. et M.C.D. chez MERISE, relations fortement facilitées par le fait que tous ces modèles utilisent le même formalisme individuel.

M.E.T. ⟶ M.C.T.

Ensemble d'opérations, synchronisations, événements et résultats, le M.C.T. constitue à la fois l'image des traitements du monde réel tels qu'ils sont perçus par l'organisation et la synthèse des représentations que se font de ces traitements les différents programmes, processus ... c'est-à-dire ce que l'on pourrait désigner par "modèles externes de traitements" M.E.T.

Ces M.E.T. sont eux aussi des ensembles d'opérations, synchronisations, événements et résultats qu'il est facile de rattacher entre eux grâce au concept de"synchronisation". Si on respecte la logique d'enchaînement des M.E.T., le suivi des opérations, on obtient ainsi le M.C.T.

Pour 'NAMUR'

M.E.D. (ou sous-schémas conceptuels des données S.S.C.D.) ⟶ S.C.D.(1)

Contrairement à ce qui est proposé par MERISE, les M.E.D. ou S.S.C.D. de 'NAMUR' ne s'inspirent pas d'un M.C.B. préalablement construit à partir du réel perçu de l'organisation.
Chacun d'eux est une structure de données propre à un traitement particulier (application, phase, (fonction)) ou à une classe d'acteurs de l'organisation.
Ils sont exprimés suivant le modèle entité-association décrit au chap. II, modèle qui servira aussi à la définition du schéma (modèle) conceptuel des données.

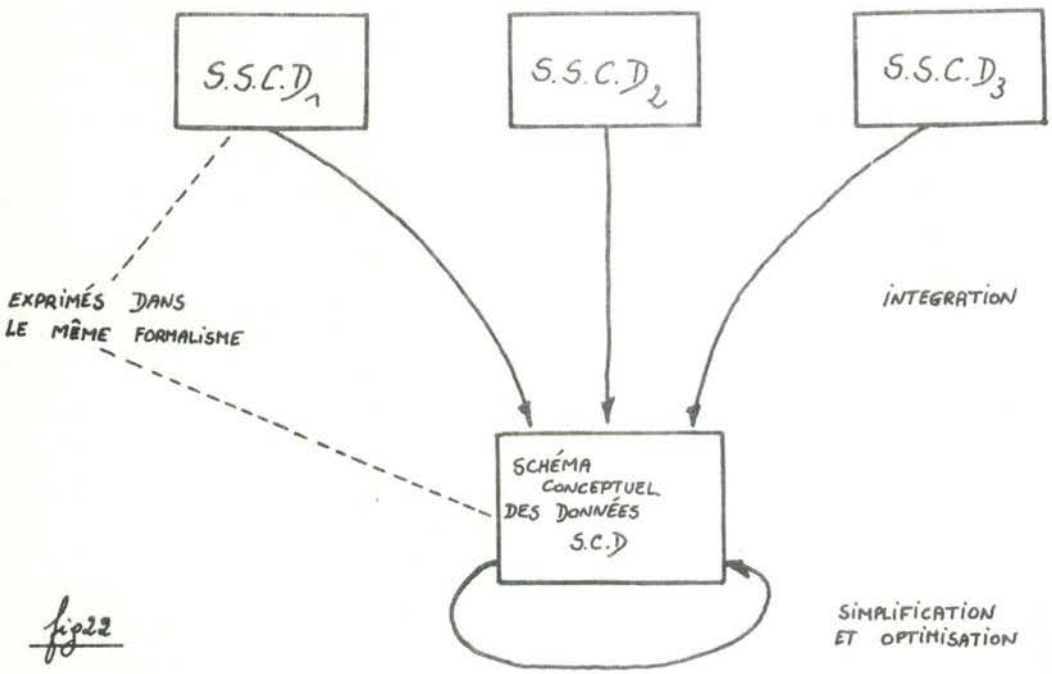


fig22

Le schéma conceptuel des données S.C.D. est obtenu par intégration des différents S.S.C.D.
Certaines simplifications et optimisations peuvent être effectuées sur le schéma obtenu en regroupant les S.S.C.D.
En effet, le S.C.D., comme d'ailleurs toute structure de données exprimée à l'aide du modèle entité-association, doit être

(1) Il s'agit ici d'une proposition méthodologique non définitive.

mis sous forme canonique. Cette dernière est caractérisée par :
- l'absence d'homonymie (pour les entités, les associations et les propriétés);
- l'absence de redondance :
  - synonymie,
  - propriétés dérivées (calculées),
  - propriétés redondantes avec une association,
    associations redondantes car dérivables par transitivité.

Passage au sous-système des traitements.

Nous avons vu dans la modélisation conceptuelle des traitements la hiérarchie proposée pour la structuration des traitements : sous-système - application - phase - fonction.

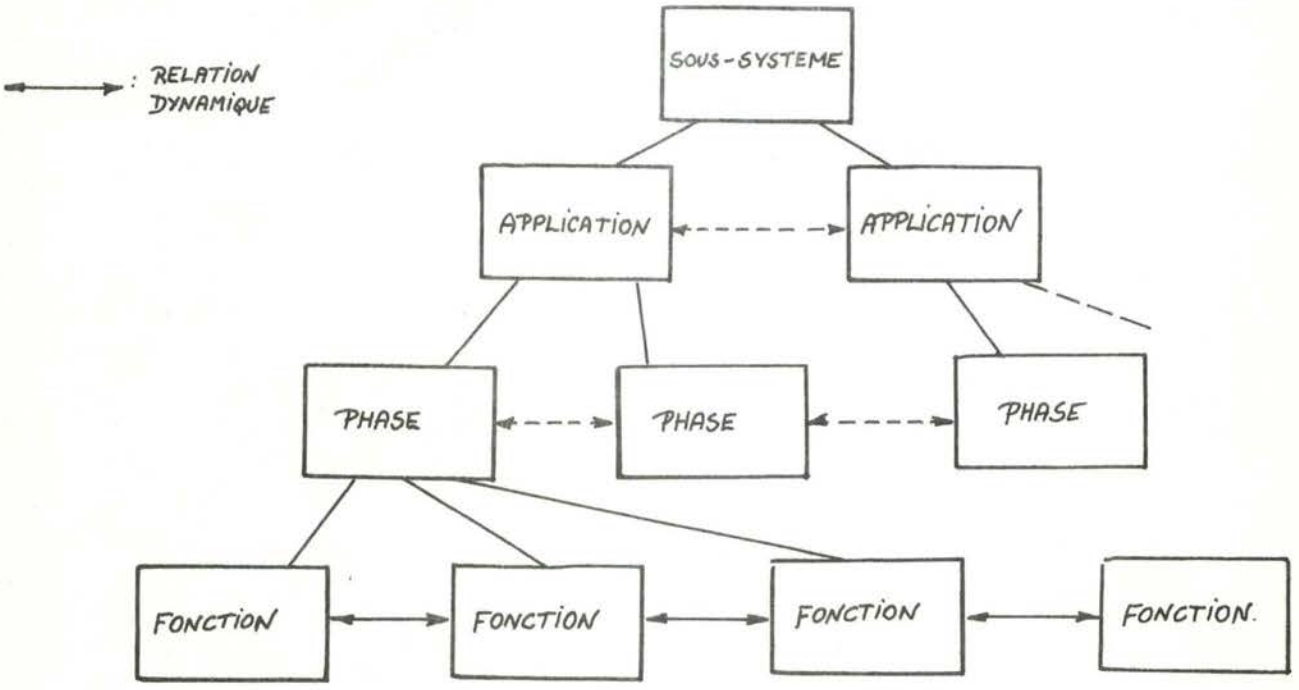


fig 23.

Le sous-système des traitements (équivalent du M.C.T. de MERISE) regroupe l'ensemble des applications et donc des phases et des fonctions. Son élaboration ne présente en principe aucune difficulté car on dispose des informations nécessaires à ce regroupement dans le réel perçu et il suffit 'd'agencer' les différents types de traitement afin d'obtenir un 'graphe' d'activité de l'organisation.

qui respecte l'enchaînement réel de ces opérations. Il faut cependant signaler que s'il est possible de synchroniser les traitements de type 'fonction' à l'aide des relations dynamiques, rien ne nous permet de le faire, à l'heure actuelle, avec les phases, les applications et le sous-système.

III 2. MAPPING entre les niveaux conceptuel et logique de la description du S.I.

Pour MERISE

MAPPING M.E.D., M.C.D. ⟶ S.S.L., M.L.D.

Cette transformation M.C.D. (M.E.D.) ⟶ M.L.D. (S.S.L.) est totalement algorithmique (un programme a d'ailleurs été réalisé à cet effet (CONCMIL)).

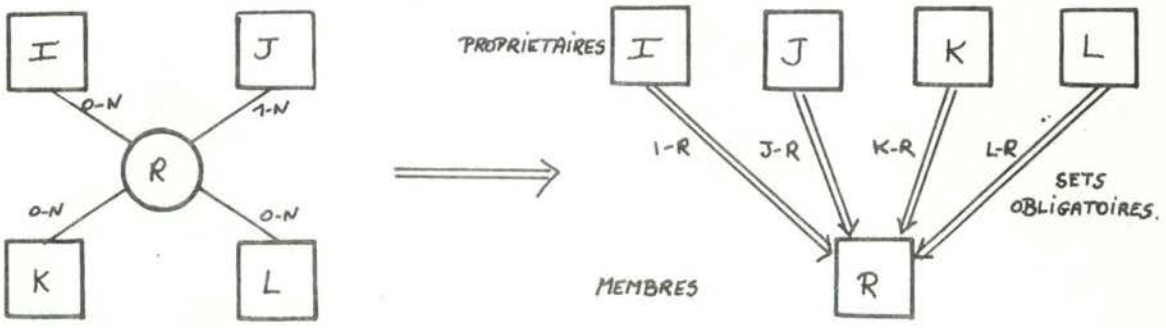
Cette traduction n'est cependant pas réversible.

Quelles sont les règles de transformation?

1) tout type d'individu devient un type de record, l'identifiant de l'individu devenant la clé du record.
2) toute propriété d'un type d'individu ou de relation devient un champ d'un type de record.
3) quant à la relation, c'est plus complexe :

- relation n-aire de cardinalité quelconque:

⟹ la relation devient un record auquel aboutissent n sets obligatoires en provenance des records découlant de la traduction des individus.



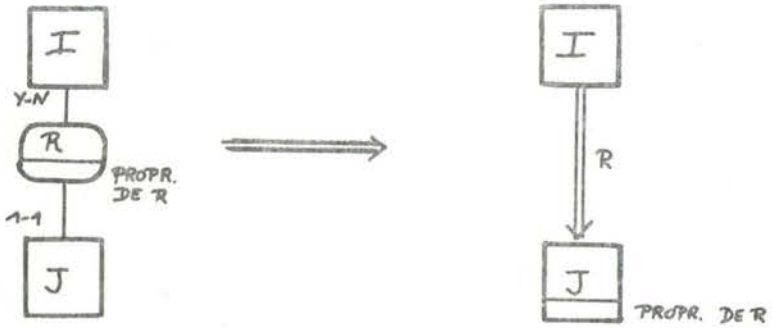
- relation binaire de cardinalité y-m, y-n (m et n>1):

⟹ la relation devient un record auquel aboutissent 2 sets obligatoires en provenance des records

découlant de la traduction des 2 individus.
(cas analogue au précédent)

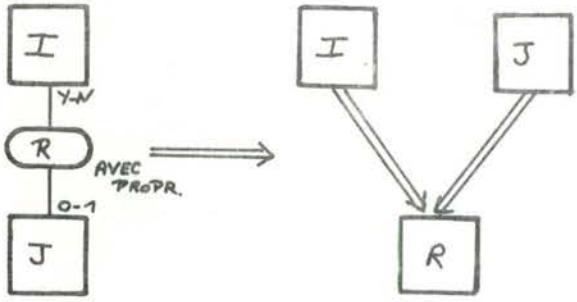
- relation binaire de cardinalité y-n, 1-1 (n > 1) :

⟹ la relation devient un set obligatoire et ses éventuelles propriétés sont transférées dans le record membre.



- relation binaire de cardinalité y-n, 0-1 (n > 1) :

si R a des propriétés : la relation devient un record auquel aboutissent 2 sets obligatoires en provenance des records découlant de la traduction des individus.

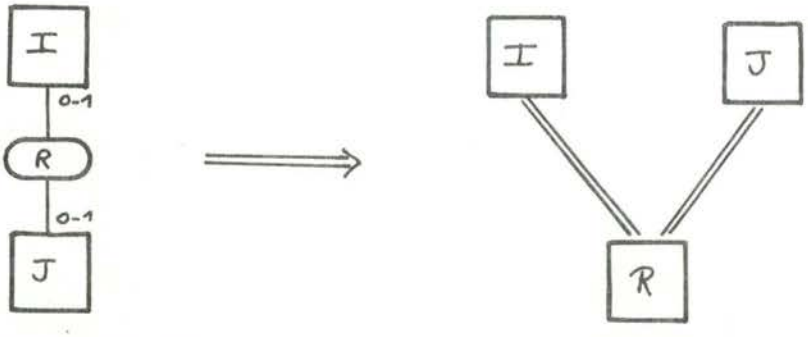


si R n'a pas de propriétés : la relation devient un set optionnel entre I et J.



- relation binaire de cardinalité 0-1, 0-1 :

⟹ La relation devient un record auquel aboutissent 2 sets obligatoires en provenance des records découlant de la traduction des individus.



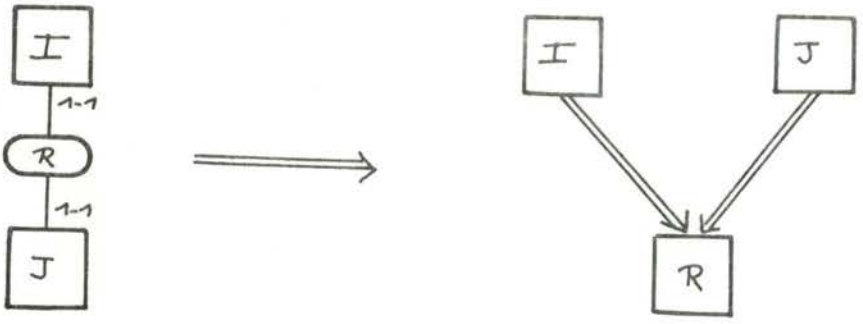
Rmq : si R n'a pas de propriétés, on devrait la traduire par un set optionnel mais quel sens lui donner?



On génèrera donc la structure ci-dessus en sachant que des simplifications sont éventuellement possibles dans le passage au modèle physique des données.

- relation binaire de cardinalité 1-1, 1-1 :

⟹ la relation devient un record auquel aboutissent 2 sets obligatoires en provenance des records découlant de la trduction des individus.



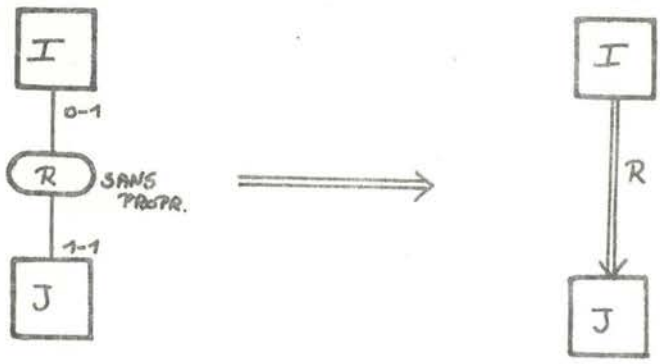
Rmq : si R n'a pas de propriétés, on aurait pu envisager un set obligatoire entre les 2 records I et J mais comment l'orienter?



On génèrera la structure énoncée ci-dessus en signalant que l'on pourra éventuellement effectuer des simplifications lors du passage au M.P.D.

- relation binaire de cardinalité 0-1, 1-1 :

⟹ on prendra comme hypothèse que la cardinalité 1-1 est plus forte que la cardinalité 0-1 et on traduira la relation par 1 set obligatoire entre les records I et J.



Rmq : si R a des propriétés ⟹ ces propriétés se retrouveront comme champs dans le record I ou le record J et on générera la structure ci-dessus (cas de R sans propriétés).

Rmq : Transformation des M.E.D. en S.S.L.

Les modèles externes peuvent avoir une structure différente mais cohérente avec celle retenue pour le M.C.D., cette cohérence étant vérifiée lors de la validation du M.C.B.D.
Il est toutefois nécessaire avant d'effectuer le mapping pour accéder au niveau logique, d'identifier les sous-schémas conceptuels des données correspondant aux M.E.D. et de leur appliquer ensuite les règles de transformation que nous avons énoncées ci-dessus afin d'obtenir les sous-schémas logiques de données (S.S.L.), sous-ensembles stricts du M.L.D.

MAPPING M.C.T. - M.L.T.

Le passage du M.C.T. au M.L.T. consiste :

- à reprendre les opérations du M.C.T. et à les classifier en :
  - procédure fonctionnelle,
  - phase,
  - tâche;
- à définir, à partir des règles de gestion du niveau conceptuel, les traitements qui s'exercent sur les données du système. C'est d'ailleurs au niveau logique des traitements que MERISE proposera et utilisera un ensemble de primitives d'accès à la B.D. (chap. IV 2.);
- à répartir ces différentes opérations entre les acteurs et le matériel (machines) de l'organisation; hommes et machines étant éventuellement regroupés en postes;

- à détailler la composition des événements / résultats, blocs logiques d'entrée / sortie;
- à décrire le graphisme d'un ou plusieurs événements / résultats : les grilles.

Il s'agit donc ici de reprendre le M.C.T. et d'y inclure toutes les considérations d'ordre organisationnel ainsi que les divers moyens logiques de réalisation.

Le passage au niveau logique des traitements n'est évidemment pas algorithmique mais certains outils peuvent être développés afin de faciliter la tâche du concepteur.

Pour 'NAMUR'

MAPPING S.C.D. (S.S.C.D.) ⟶ S.A.L. (S.S.A.L.)

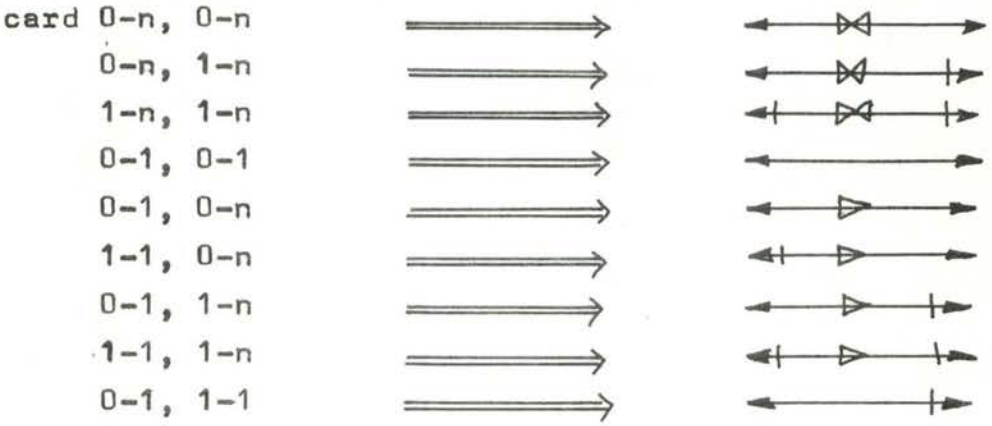
Nous allons donner ici les règles de transformation du schéma conceptuel des données (ou d'un sous-schéma conceptuel des données) en schéma des accès logiques (ou en sous-schéma des accès logiques) . Ces règles sont les suivantes :

a) à tout type d'entité est associé un type d'article, chaque entité étant représentée par un article;

b) à tout attribut est associé un item;

c) pour les relations, on aura :

relation binaire sans propriété ⟹ 2 relations d'accès, inverses l'une de l'autre quant aux cardinalités, elles seront transformées de la façon suivante :

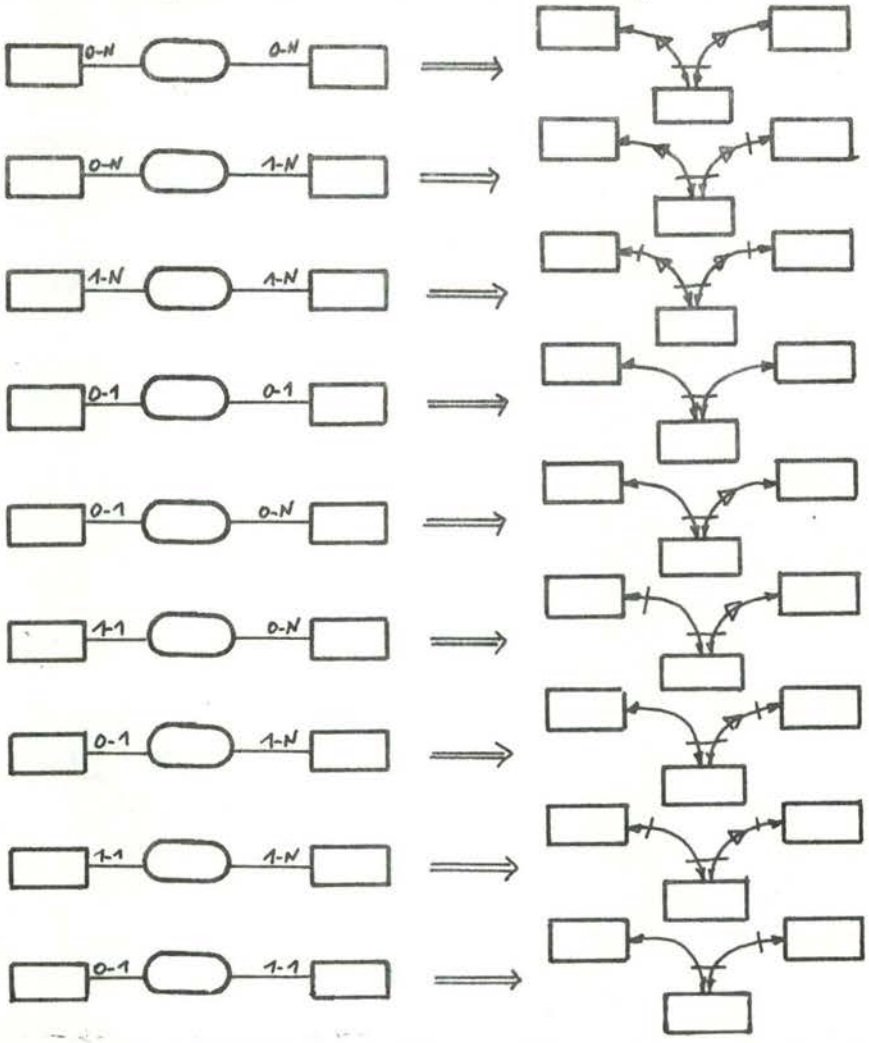


relation binaire avec propriétés ⟹ le type d'association devient un type d'article relié à deux autres types d'articles, traduction des deux types d'entités, par

des relations d'accès, inverses l'une de l'autre .

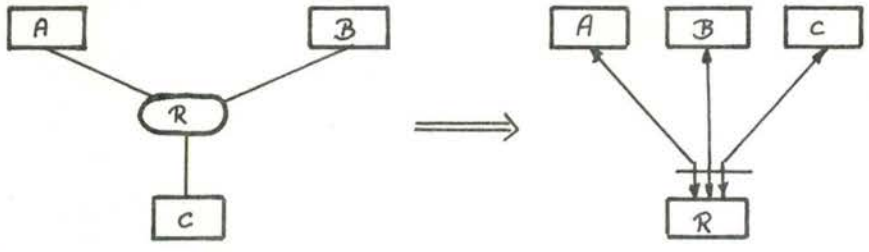


Pour ce qui est des cardinalités, on aura :



relation n-aire ⟹ le type d'association devient un type d'article relié à n autres types d'articles, traduction des n types d'entités, par des relations d'accès, inverses l'une de l'autre.

ex : relation 3aire



Pour les cardinalités : cfr considérations sur les relations binaires avec propriétés.

d) un accès séquentiel est prévu pour tout type d'article

Rmq : Ici les S.S.C.D. - M.E.D. étant des sous-ensembles stricts du S.C.D. (contrairement aux M.E.D. de MERISE), on peut leur appliquer directement les règles de transformation ci-dessus afin d'obtenir les sous-schémas d'accès logiques.

Passage de niveau conceptuel des traitements au niveau logique des traitements.

Le niveau logique des traitements reprend le niveau conceptuel en le détaillant. En effet, les traitements de type "fonction" sont explicités sous forme d'algorithmes qui utilisent notamment des primitives d'accès pour leur définition (nous reprendrons ce point au chapitre suivant).
De plus, les spécifications dynamiques seront transformées en algorithmes de contrôle des flux des messages.
C'est aussi au niveau logique que l'on spécifiera les moyens de réalisation humains et techniques qui seront affectés aux différents traitements. On répartira ainsi les processeurs et on détaillera les ressources utilisées ainsi que leurs modalités d'utilisation.
Le passage au niveau logique n'est pas implémentable mais on peut envisager, tout comme pour MERISE, des outils qui faciliteraient la tâche du concepteur en ce domaine.

III 3. Passage au niveau physique (p.m.).

Ce passage consiste à reprendre le modèle logique des données ou le schéma des accès logiques et de les transformer en modèle ou schéma interne en prenant en compte un certain nombre de contraintes physiques : moyens physiques de réalisation et logiciels (notamment S.G.B.D.).
On doit obtenir à la fin de cette étape, la B.D. complète de l'organisation. Au point de vue des traitements, les différents algorithmes de niveau logique vont être traduits en programmes écrits dans un langage de programmation exécutable, programmes qui s'exécuteront sur la B.D. que l'on aura obtenue.

III 4. Cohérence entre les modèles de données et les modèles de traitements.

A chacun des niveaux de description du S.I. (conceptuel, logique (physique)), il y a une cohérence à respecter entre les modèles de données et les modèles de traitements. C'est ainsi qu'à chaque niveau :

- les données doivent constituer les ressources pour les traitements,
- les traitements sont des transformateurs pour les données.

Ce n'est que si cette cohérence est vérifiée que l'on pourra parler de modèles-schémas conceptuel - logique - physique du système et non plus seulement de modèles et schémas conceptuel - logique - physique { - de données - de traitements.

Examinons donc quelles sont à chaque niveau, les relations entre données et traitements.

Pour MERISE

Relations de niveau conceptuel

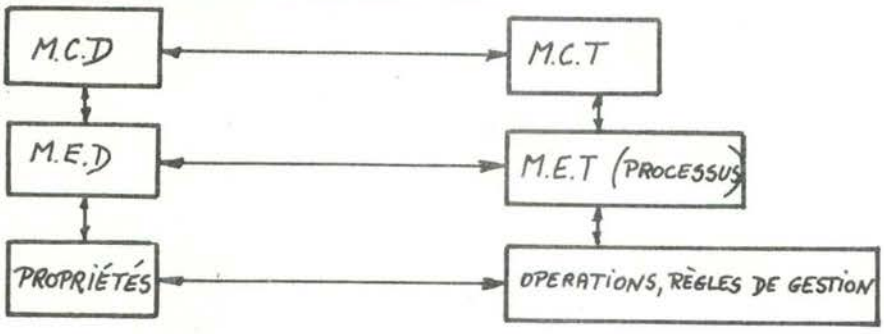


fig 24.

Relations de niveau logique

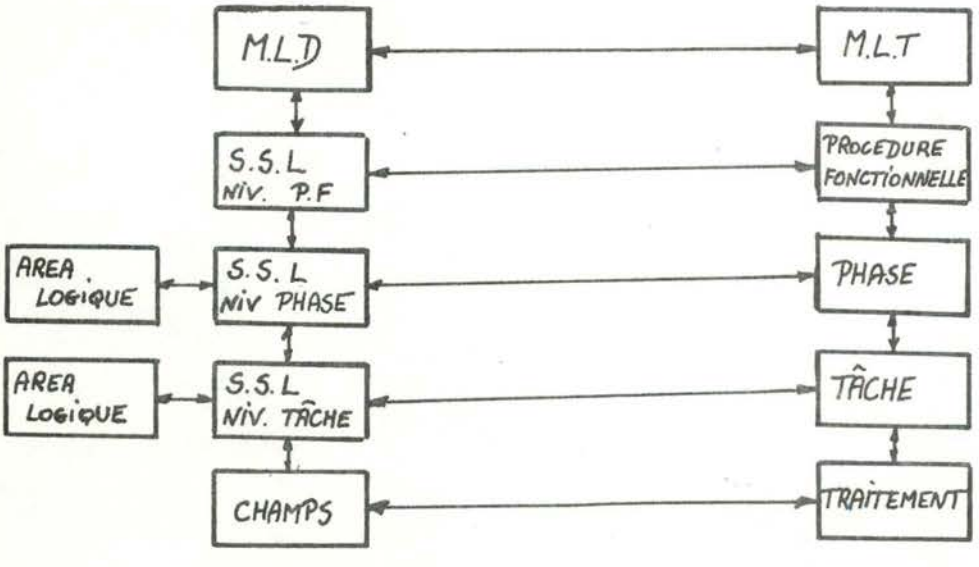


fig25

Pour 'NAMUR'

Relations de niveau conceptuel

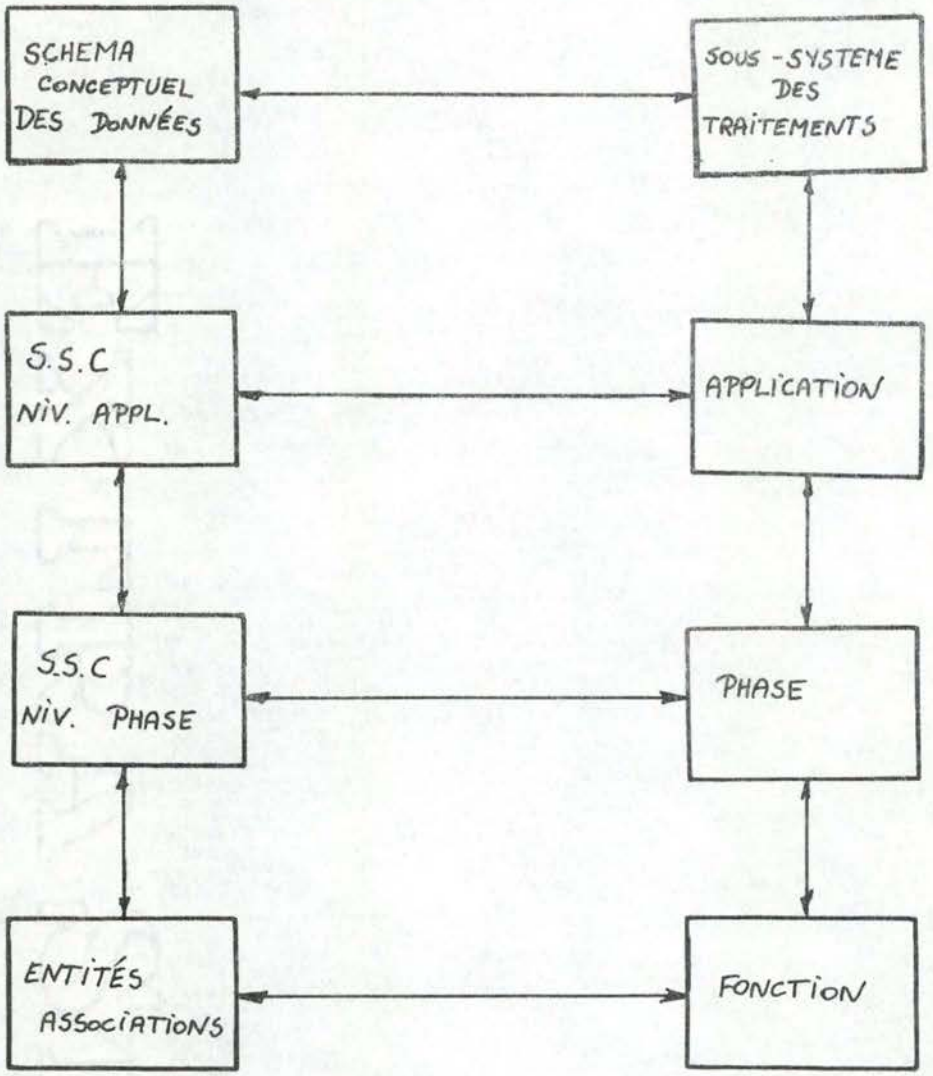


fig 26

Relations de niveau logique

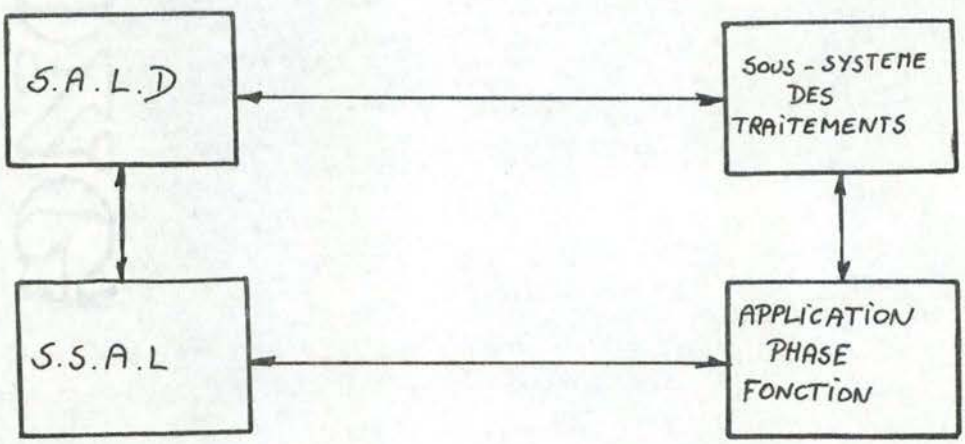


fig 27

Chap. IV. Comparaison des méthodes proposées par MERISE et 'NAMUR'.

Nous allons étudier dans ce chapitre les méthodes proposées par MERISE et 'NAMUR'. Ces 2 méthodes, comme toute méthode d'ailleurs ont pour but de fournir un cadre général, un support continu pour l'action. Elles doivent permettre d'envisager tout type de problème qui peut se poser à l'organisation, même des situations nouvelles qui n'ont jamais été traitées, et proposent un découpage du processus de conception du S.I. en étapes cohérentes : les résultats d'une étape étant le point de départ de l'étape suivante. Elles se veulent être plus un guide qu'un carcan, ne prescrivant que quelques éléments primordiaux stables et simples à comprendre par tout le monde, et laissent pour le reste une large part à l'initiative et à l'intelligence des concepteurs du futur système.

Les méthodes MERISE et 'NAMUR' se basent sur les niveaux de description que nous avons exposés au chap.I (cfr. I.12) à savoir : le réel perçu, le niveau conceptuel (modèles conceptuel et externes), le niveau logique et le niveau physique.

Le réel perçu se présente sous la forme d'une description assez détaillée de l'organisation, description souvent littéraire (rapports d'experts, comptes rendus d'interviews,...) de son activité, de ses choix et objectifs. Chacun des autres niveaux - conceptuel, logique, physique - utilisera, pour son expression, deux modèles : un pour les données et un pour les traitements; ces modèles ont été décrits au chap.II. DE plus, des processus de passage d'un niveau à l'autre, ainsi que des règles de cohérence entre données et traitements, ont été retenus pour chacune des deux méthodes (chap.III). Certaines phases dans la conception du S.I. peuvent être formalisées et même faire l'objet d'algorithmes, d'autres cependant sont et ne peuvent être que de nature heuristique : au plus peut-on alors construire des outils qui aideront le concepteur dans sa tâche(la partie relative aux outils développés dans le cadre des deux méthodes fera l'objet du chapitre suivant.

Nous allons donc maintenant étudier en détail les méthodes MERISE et 'NAMUR', les différentes étapes qu'elles proposent ainsi que leur articulation. Certains points ont cependant déjà été étudiés aux chapitres précédents; nous nous contenterons alors

de les mentionner.

IV. 1. Méthode MERISE : les étapes et leur articulation.

IV. 11. Elaboration des modèles de niveau conceptuel.

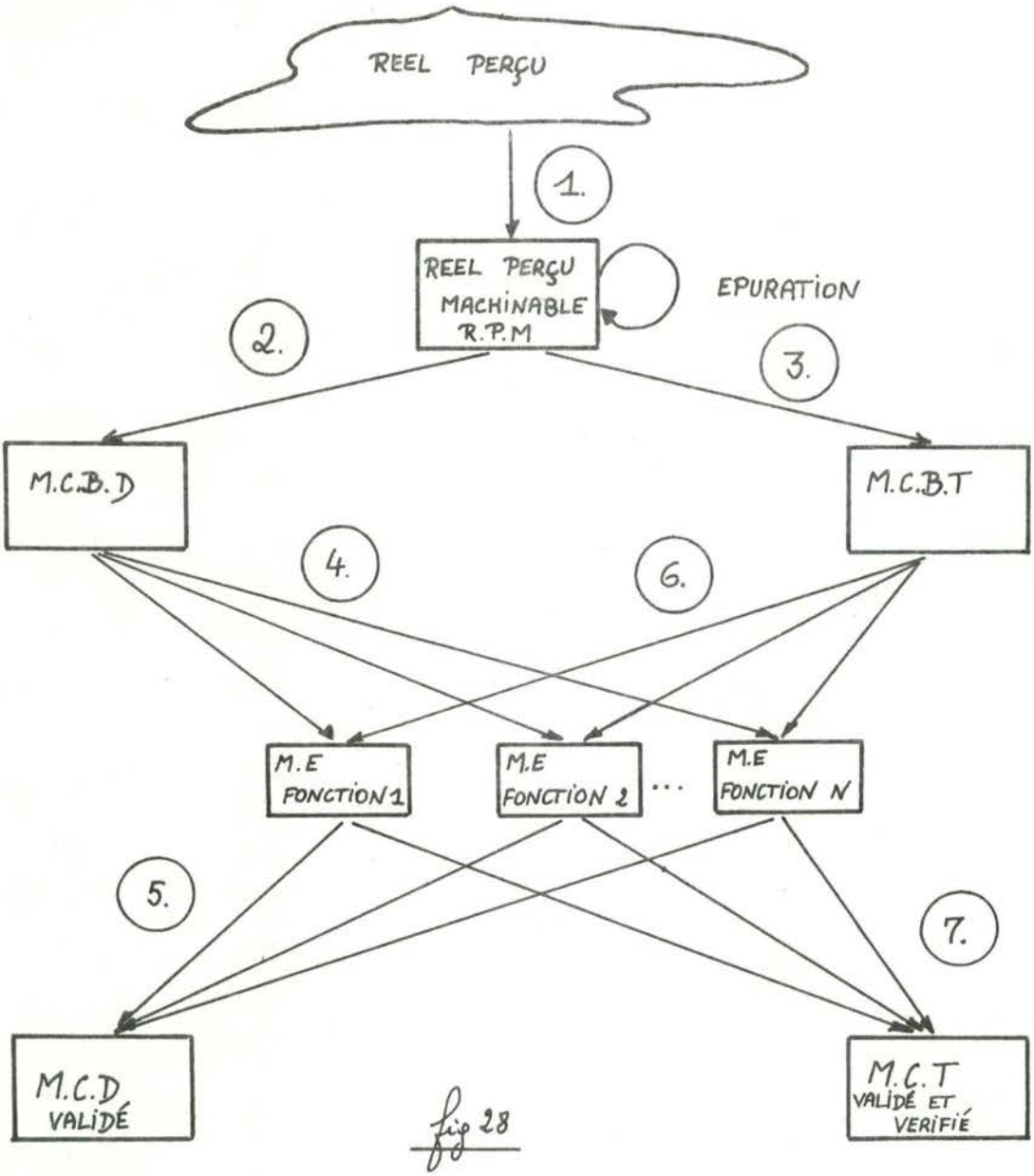


fig 28

(1.) Passage du réel perçu (R.P.) au réel perçu machinable (R.P.M.).

Le R.P. contient toutes les informations dont on a besoin pour élaborer le S.I. de l'organisation. Il se caractérise par :

- une liste d'informations élémentaires (ex. : nom de client, n° de produit,...),

(1) Ces informations sont exprimées suivant les différents langages outils en vigueur dans l'organisation, langages naturels utilisés par les différentes classes d'utilisateurs : bordereaux, fiches,...

- une liste de contraintes d'intégrité qui en précisent le sens (ex : une commande émane d'un et un seul client),
- une liste d'opérations, de résultats et de règles de gestion,
- un ensemble de quantifications,
- ...

Ces informations proviennent :

- d'anciennes fonctions automatisées : programmes ou applications existantes qui constituent une perception particulière qu'il faut analyser en tant que réel perçu particulier et qu'il faudra faire fonctionner sur le S.I. que l'on va construire;
- de fonctions à automatiser : programmes ou applications projetés dont on peut déjà dégager certaines informations (entrées-sorties);
- de la description de l'organisation : qui tente de présenter une structuration stable et correcte de l'organisation. On identifie ainsi :
  - les sous-systèmes de l'organisation,
  - les transactions le long des flux physiques (logistique, monétaire, personnel, actifs) entre deux sous-systèmes ou entre un sous-système et un élément actif identifiable de l'environnement.

Le R.P.M. reprend toutes les informations du R.P. et les met sous une forme 'machinable', c'est-à-dire :

- que la liste des propriétés (informations élémentaires) est épurée, notamment sans polysèmes ni synonymes (1),
- que le vocabulaire et l'orthographe, tant pour les données que pour les traitements, sont normalisés,

---

(1) On appelle polysème deux informations ayant le même nom et des sens différents.
On appelle synonyme deux informations ayant le même sens et des noms différents.

Des outils facilitant le détection de polysèmes et de synonymes ont été développés dans le cadre de la méthode MERISE. (cfr. chap. V). Ils sont basés notamment sur l'utilisation de mots-clés.

- que les codifications sont établies,
- que les contraintes d'intégrité et les règles de gestion du R.P. sont exprimées à l'aide du vocabulaire du R.P.M.

2. Passage au modèle conceptuel brut des données (M.C.B.D.).

Il s'agit maintenant d'exprimer, pour les données, le R.P.M. dans un formalisme adapté au niveau conceptuel et MERISE a choisi d'utiliser pour cela le modèle individuel, modèle que nous avons décrit au chapitre II.

Quant à la technique qui permet d'effectuer ce passage au M.C.B., elle relève de la famille des techniques de conception assistée par ordinateur qui permettent, à partir de certaines règles de construction propre au formalisme choisi, de laisser le concepteur imaginer des solutions tout en vérifiant à chaque instant que la solution choisie ne contredit pas les règles fixées. On a donc développé un outil de conception de modèles individuels (CAOMI) qui aide à la définition des individus et des relations à partir de la liste d' informations du R.P.M. Une fois le modèle construit, le concepteur peut alors définir les cardinalités et les contraintes d'intégrité sur les relations. Des outils (cfr. chap.V) en vérifieront la cohérence et effectueront éventuellement la décomposition de certaines relations en relations de dimension inférieure.(2)

A l'issue de cette étape, on dispose du M.C.B.D., complètement documenté, vérifié et pouvant être édité ou dessiné à la demande.

3. Passage au modèle conceptuel brut des traitements (M.C.B.T).

Le R.P.M. contient un ensemble d'informations relatives aux traitements qui doivent être exécutés par le futur S.I. On dispose en effet de listes d'événements, de résultats, d'opérations, de règles de gestion (1), ainsi que des indications sur les sous-systèmes de l'organisation (données organisationnelles qui nous serviront surtout lors du passage au niveau logique des traitements).

Toutes ces informations sont agencées grâce au modèle conceptuel des traitements (1), de façon à obtenir le M.C.B.T. :

---

(1) cfr. chap.II : M.C.T.

(2) Cette décomposition s'avérera intéressante lorsque nous passerons au niveau logique des traitements .

- les opérations composées d'un certain nombre de règles de gestion, sont regroupées en procédures fonctionnelles,
- les événements sont affectés, éventuellement après synchronisation, aux différentes opérations qu'ils déclenchent,
- les résultats sont rattachés aux opérations qui les produisent, souvent en fonction de règles d'émission.

Cet agencement est possible car le R.P. contient, comme nous l'avons vu, une 'photographie' plus ou moins précise et complète de l'organisation.

(4.) Elaboration des M.E.D. à partir du M.C.B.

En utilisant le vocabulaire - mais pas nécessairement la structure - du M.C.B., on élabore les différents M.E.D. suivant les règles énoncées au chap.III (III.1 : M.E.D.⟶M.C.D.). Chaque M.E.D est souvent propre à une fonction, automatisée ou non. Il utilise, pour la définition de sa structure et de son contenu, les informations que l'on peut notamment trouver dans les événements et les résultats de la fonction, informations que l'on retrouve d'ailleurs pour la plupart dans le M.C.B. sous forme de propriétés d'individus ou de relations.

(5.) Validation du M.C.B. par les M.E.D. et élaboration du M.C.D.

Ce point a été développé au chap.III (III.1 : M.E.D⟶M.C.D.).

(6.) Identification des M.E.T.

Il s'agit d'élaborer à cette étape les M.E.T. - ensemble d'une ou plusieurs procédures fonctionnelles - et ce en s'aidant du M.C.B.T. . Ces M.E.T. représentent en fait les fonctions exercées dans l'organisation et ils servent notamment à la construction des M.E.D. qui leur sont associés. Bien qu'ils s'inspirent du M.C.B.T., (ils utilisent d'ailleurs le même moyen d'expression : le modèle conceptuel des traitements (chap. II : M.C.T.)) ,les M.E.T. n'en constituent pas toujours pour autant des sous-ensembles stricts : leur structure peut être différente et une étape de validation s'impose pour vérifier la cohérence entre les M.E.T. et le M.C.B.T.

(7.) Validation du M.C.B.T.

Il faut s'assurer que tout ce qui apparaît au niveau des M.E.T. peut se retrouver, d'une manière ou d'une autre, dans le M.C.B.T. C'est ainsi que :

- les opérations qui apparaissent dans les M.E.T. sont des (sous-) opérations du M.C.B.T.,
- les événements et les résultats des M.E.T. doivent se retrouver soit en tant que tels, soit sous forme de "sous événements" ou de "sous résultats" du M.C.B.T.,

  Ex. :

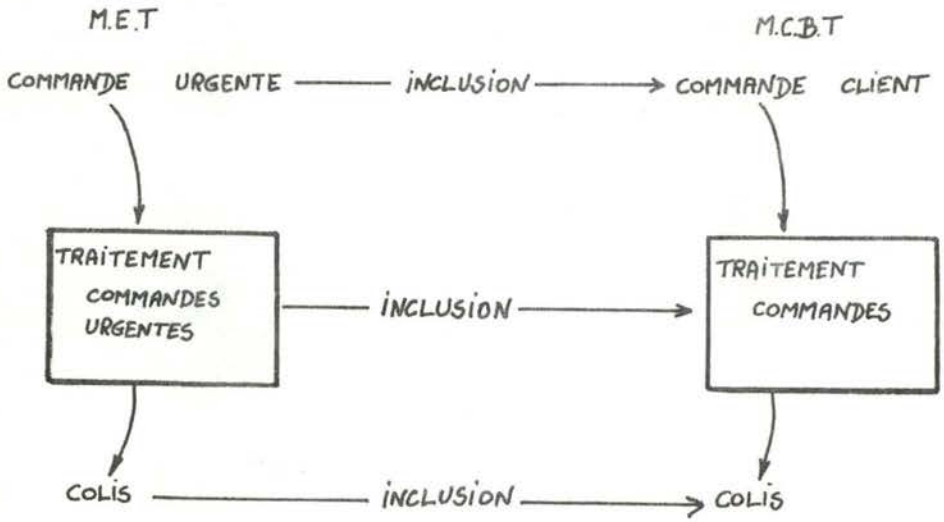


fig 29.

- les règles de gestion des M.E.T. doivent se retrouver dans le M.C.B.T. ,
- de plus, l'événement du M.C.B.T. qui correspond à un événement d'un M.E.T. doit déclencher une opération qui englobe l'opération déclenchée par l'événement du M.E.T.

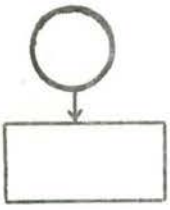
Il s'agit ici plutôt de règles de construction de M.E.T. plutôt que de règles de validation du M.C.B.T. . En effet, ce dernier a été construit à partir d'informations provenant des fonctions mêmes et il est donc, de par sa construction, pratiquement toujours validé. Le M.C.B.T. doit cependant encore répondre à un certain nombre de règles de vérification avant de pouvoir être considéré comme M.C.T. validé et vérifié. Ces règles sont :

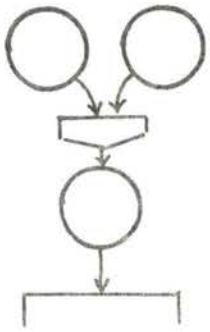
- Vérification des opérations :

  1) tout type d'opération doit avoir un et un seul type d'évé-

nement en entrée

Ex. :





2) tout type d'opération a au moins un type de résultat en sortie. Ce résultat est délivré en fonction d'une expression dont les facteurs proviennent soit des propriétés de l'événement en entrée, soit des propriétés du modèle des données.
3) dans tous les cas, une occurrence d'opération délivrera au moins une occurrence d'au moins un type de résultat.
4) toute opération appartient à un processus et un seul; les processus sont donc disjoints.

• Vérification des événements et résultats.

5) définitions :

événement externe : produit par un acteur vers une opération ou synchronisation (éventuellement plusieurs),

événement interne : produit par une synchronisation vers une seule opération,

résultat externe : produit par une opération vers un acteur,

résultat interne : produit par une opération vers une synchronisation (le cas échéant, une opération) ou plusieurs.

6) lorsqu'une synchronisation ou une opération est activée, il y a consommation d'une occurrence (éventuellement n) du ou des types d'événement/résultat qui provoquent cette activation; ceci peut conduire à regénérer une occurrence d'un type d'événement/résultat ou à produire un même type de résultat sous la forme de plusieurs occurrences identiques (on dé-

finira ainsi la cardinalité d'un événement ou d'un résultat (1)). D'autre part, on devra s'assurer que toute occurrence produite est bien consommée.

• Vérification des synchronisations.

7) une synchronisation a en entrée n ($n \geqslant 2$) type(s) d'événement externe ou type(s) de résultat interne; elle a en sortie un type d'événement interne unique.
8) l'expression booléenne d'une synchronisation ne peut pas :
   -être toujours égale à 1 : pas d'attente, on parlera alors de 'pseudo-synchronisation',
   -être toujours égale à 0 : l'opération correspondante ne serait jamais activée : résultat(s) non-atteignable(s).
9) lorsqu'un type d'événement externe (ou un résultat interne) participe à plusieurs synchronisation(s) ou opération(s), on dit qu'il y a conflit possible pour ou sur ce type d'événement; le conflit est résolu si les conditions d'entrée sont exclusives (cas des synchronisations) ou si le type d'événement/résultat est muni d'une cardinalité convenable : autant d'occurrences du type d'événement/résultat que d'opération(s)/synchronisation(s) en conflit.
10) lorsqu'une synchronisation est activée par un résultat dont elle déclenche elle-même la production à travers une ou plusieurs opérations, il y a cycle; tout cycle doit être détecté; il faut expliciter de plus ses conditions d'amorçage et d'arrêt.

---

(1) la cardinalité de l'événement -type E ou du résultat-type R en entrée de la synchronisation S ou en sortie de l'opération O est le nombre d'occurrences de E ou R qui interviennent en entrée de S ou en sortie de O.

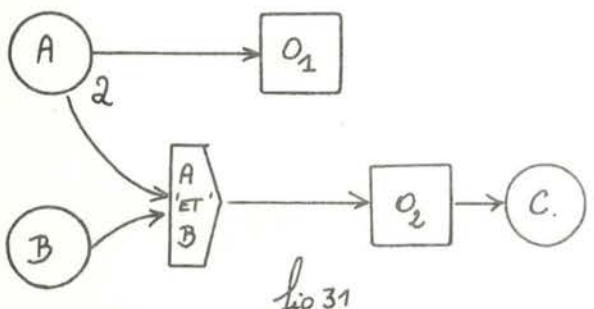


fig 31

- il existe 2 occurrences de l'événement A; cela évite un conflit entre la synchronisation S, et l'opération O,
- l'opération O produit le résultat C en 2 exemplaires.

IV.12. Elaboration des modèles de niveau logique.

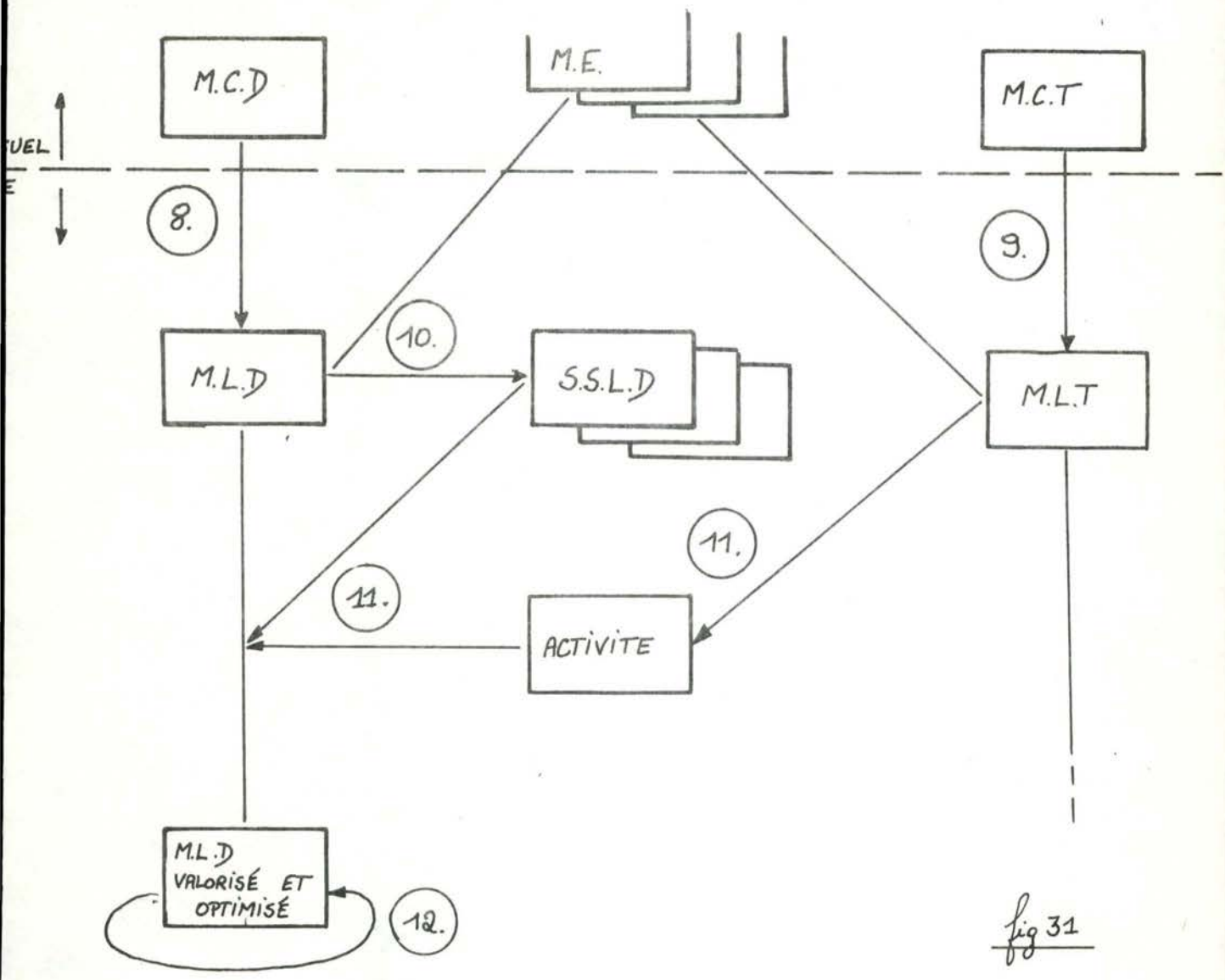


fig 31

(8.) Mapping M.C.D.→M.L.D.

Cette traduction du M.C.D. en M.L.D. a été décrite au chap.III(III.2)

(9.) Mapping M.C.T.→M.L.T.

Ce point a également été présenté au chap.III(III.2)

(10.) Traduction des M.E.D. en S.S.L.D.

Comme nous l'avons vu précédemment, les M.E.D. qui correspondent à des fonctions particulières de l'organisation, peuvent avoir une structure différente de celle qui est retenue pour le M.C.D. : une étape de validation est d'ailleurs nécessaire afin de vérifier la

cohérence entre ces modèles. C'est pourquoi, il faut d'abord identifier le 'sous-schéma conceptuel' qui correspond au M.E.D. envisagé avant de lui appliquer les règles de MAPPING entre les niveaux conceptuel et logique(III.2). On obtiendra ainsi le S.S.L.D.(sous-schéma logique des données) qui est un sous-ensemble strict du M.L.D.

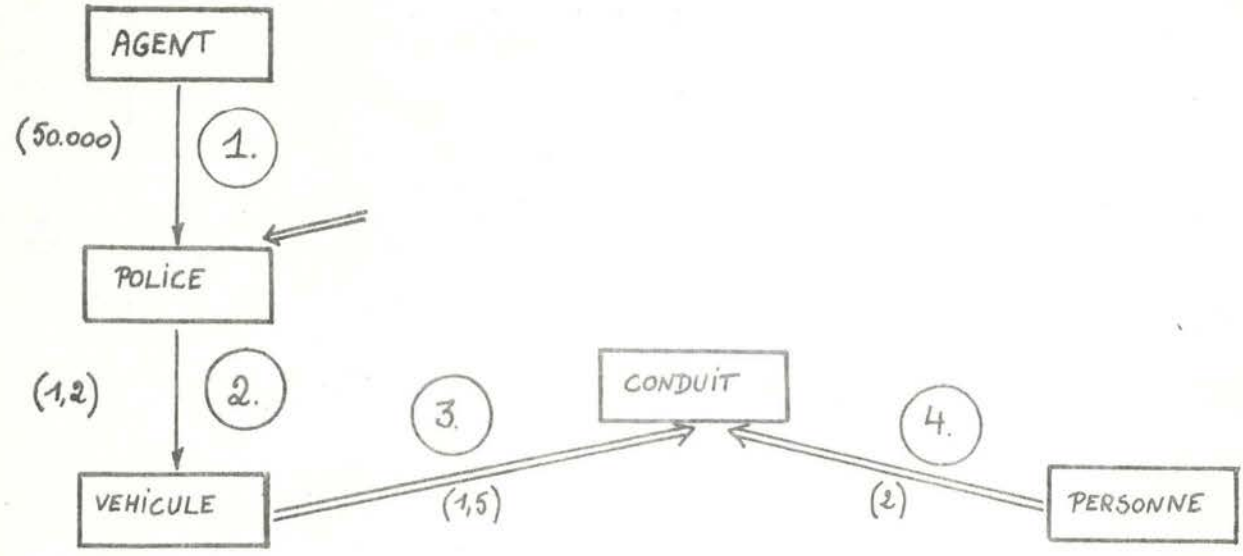
(11.) Prise en compte de l'activité sur la future base de données.

Cette activité est le reflet des actions effectuées par les différentes fonctions (M.E.T.) sur les S.S.L.D. On essayera donc d'exprimer les opérations qui s'exercent sur la structure des données décrite par les S.S.L.D., en utilisant un ensemble de primitives d'accès : actions élémentaires que l'on multipliera par la fréquence d'activation des fonctions par période afin d'obtenir une mesure de l'activité sur le M.L.D.

Primitives d'accès :

- $\#R$: accès à un record connaissant la valeur de sa clé.
- R^+ : ajout d'une occurrence d'un type de record.
- R^- : suppression d'une occurrence d'un type de record auquel on aura accédé auparavant.
- R^m : modification d'une occurrence d'un type de record, c.-à-d. modification de la valeur d'un ou plusieurs champs d'une occurrence d'un type de record.
- S, S^{-1}, S^F : parcours d'une occurrence d'un type de set,
  - S : accès maître $\longrightarrow 1^{er}$ membre
  - S^{-1} : accès membre $\longrightarrow$ maître
  - S^F : accès membre $\longrightarrow$ membre suivant.
- S^+ : ajout d'une occurrence de type record 'membre' dans une occurrence d'un type de set
- S^- : suppression d'une occurrence de type record 'membre' dans une occurrence d'un type de set.

Grâce à ces primitives, on va pouvoir décrire maintenant l'activité sur les différents S.S.L. Un exemple va nous permettre de mieux saisir la façon de procéder. Il s'agit de montrer ce qu'il se passe au niveau du M.L.D. (et donc ce qui se passera plus tard au niveau de la B.D.), chaque fois qu'il y a création d'une nouvelle police d'assurance. Voici le S.S.L. associé (S.S .L. simplifié, bien entendu) :



( ) numérotation des sets

( ) cardinalité moyenne des sets (information du R.P.)

⟹ point d'entrée dans le S.S.L. (fourni par le concepteur)

A partir de ce schéma, on peut dresser le tableau suivant :

	Primitives	Act. unit.	Act. tot.
création d'une police	police$^+$	1	10.000
création véhicule	véhicule$^+$	1,2	12.000
rattachement de véhicule	S_2^+	1,2	12.000
création de conduit	conduit$^+$	1,2x1,5	18.000
rattachement de conduit	S_3^+	1,2x1,5	18.000
accès à personne	#personne	1,2x1,5	18.000
création de personne	personne	1,2x1,5x½ (1)	9.000
rattachement de conduit	S_4^+	1,2x1,5	18.000
identification d'agent	#agent	1	10.000
rattachement de police	S_1^+	1	10.000

Hypothèse : par période, 10.000 créations de police.

(1) On a tenté d'accéder à 1,2x1,5 personnes (#personne) mais seules 1,2x1,5x½ sont présentes; on décide donc d'en créer 1,2x1,5x½.

• **Comment construire ce tableau de l'activité sur le S.S.L.?**

Rappel : chaque S.S.L. correspond à une fonction bien particulière dans l'organisation.

Il découle d'un modèle externe - en mise à jour (création, suppression, modification d'individus et de relations
- en consultation (uniquement de la lecture).

Pour un S.S.L. issu d'un M.E.D. en mise à jour, les primitives considérées sont : R^+, R^-, $\#R, R^m$, S^+, S^-

R^+: création d'une occurrence d'un type de record.

Point d'entrée (logique) dans le S.S.L. : le type de record dont on veut créer, au départ, une occurrence.
Ce point d'entrée doit être fourni par le concepteur.
Ex : dans l'exemple précédent, on veut créer une occurrence de police d'assurance → point d'entrée : police ⟹ POLICE

On devra considérer alors les types de sets et de records accessibles à partir du type de record dont on a créé une occurrence, voire s'il y a lieu de créer d'autres occurrences de records et de sets, d'effectuer les accès nécessaires et d'opérer certains rattachements.
Si le type de record dont on a créé une occurrence est membre d'un type de set, il faut alors accéder à une occurrence du record maître (éventuellement) la créer R^+) y relier l'occurrence de record membre, puis créer (si nécessaire) une nouvelle occurrence du type de set défini entre ces deux types de records (S^+) (création obligatoire ou facultative selon que le set est obligatoire ou optionnel).
Si le type de record dont on a créé une occurrence est maître

d'un type de set, il faut alors accéder à (voire créer) n occurrences du record membre (n = card. moyenne du set) et rattacher toutes ces occurrences en créant des occurrences du type de set défini entre les deux types de records dans le S.S.L. (accès et rattachement(s) obligatoire(s) si le set est obligatoire).

Comment calculer l'activité unitaire ?

activité unitaire	= 1 si création d'une occurrence du type de record choisi comme point d'entrée du S.S.L. = produit des cardinalités moyennes des sets à parcourir pour arriver à ce type de record* x probabilité de créer les records du type envisagé**

*: pour les types de sets parcourus dans le sens membre(s)-maître, la cardinalité moyenne est considérée comme étant égale à 1.

**: il n'est pas souvent nécessaire de créer autant d'occurrences de record que théoriquement prévu, certaines occurrences existant déjà par ailleurs et il suffit alors d'y accéder.
Cette probabilité peut être obtenue assez facilement à partir des quantifications et de certains calculs préliminaires.

#R : accès à une occurrence d'un type de record.

Il s'agit d'accéder à une occurrence d'un certain type de record connaissant la valeur de sa clé (que l'on ne spécifiera pas à ce niveau).

Cette primitive est utile 1) pour situer l'occurrence du type de record choisi comme point d'entrée du S.S.L. (pour S.S.L. issu de M.E.D. en consultation (cf. plus loin))

2) pour identifier l'occurrence de record maître correspondant à l'occurrence de record membre sur laquelle on s'est positionné

et ce, lorsque l'occurrence de set qui devrait relier ces deux occurrences de records n'a pas encore été créée (une des étapes suivantes sera souvent ce rattachement maître-membre (S^+))

3) pour identifier des occurrences de records existantes et que l'on voudrait voir rattachées à une occurrence de record maître

Comment calculer l'activité unitaire ?

activité unitaire	= produit des cardinalités moyennes des sets à parcourir pour arriver au type de records auquel on veut accéder

(pour les types de sets parcourus dans le sens membre(s)-maître, la cardinalité moyenne est considérée comme étant égale à 1)

Cette activité unitaire exprime le nombre de tentatives d'accès; cela ne veut pas dire, en effet, que le nombre de records accédés est le même.
(dans l'exemple p.8 : on tente d'accéder à 1,2x1,5 records 'Personne' mais une fois sur deux en moyenne, ces records n'existent pas⟶on aura à créer 1,2x1,5x0,5 records 'Personne'.)

R^- : suppression d'une occurrence d'un type de records.

Elle suppose que l'on aura accédé au préalable à l'occurrence de record à supprimer :

- accès direct #R
- recherche dans les différents sets impliqués.

On ne peut supprimer une occurrence de record que si elle ne sert plus à accéder à d'autres occurrences de sets et de records, eux aussi objets d'une éventuelle suppression.
La suppression d'occurrences de records entraînent souvent la suppression d'occurrences de sets. La logique à suivre est semblable à celle prise pour R^+.

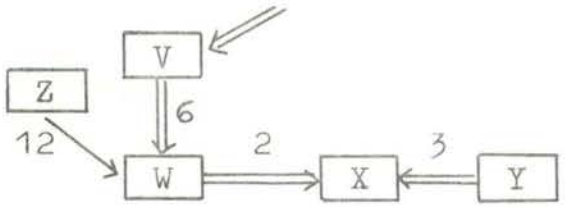
Comment calculer l'activité unitaire ?

activité unitaire = 1 si suppression d'une occurrence du record choisi comme point d'entrée
= produit des cardinalités moyennes des sets à parcourir pour arriver à ce type de record*
X probabilité de supprimer les records du type envisagé**

*: pour les types de sets parcourus dans le sens membre(s)-maître, la cardinalité moyenne est considérée comme étant égale à 1.

**: il n'est pas toujours nécessaire de supprimer toutes les occurrences de records (et de sets) liées à l'occurrence du record que l'on veut supprimer au départ.

Ex.



$V^- : 1$

$W^- : 6$

$Z^- : 6$

$X^- : 6 \times 2 = 12$

$Y^- : 6 \times 2 \times 0,5 = 6$

je décide de ne supprimer les occurrences de Y qu'à raison de 1 sur 2.

R^m : modification d'une occurrence d'un type de record.

Après y avoir accédé, on modifie l'occurrence du record. Cette modification est tout-à-fait locale c.-à-d. qu'elle n'a aucune répercussion sur les autres occurrences de sets et de records.

Activité unitaire.

activité unitaire = 1

S^+ : ajout d'une occurrence d'un type de set

Cela suppose que l'on ait accédé au préalable à l'occurrence du record maître et à l'occurrence du record membre (éventuellement création de ces occurrences), occurrences que l'on veut relier par une nouvelle occurrence du set défini entre ces deux types de records.

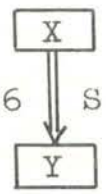
Rmq : chaque fois que l'on crée une occurrence de membre de set obligatoire, on doit ajouter une nouvelle occurrence de ce set. Il n'en est pas de même lorsqu'on crée une occurrence de membre de set facultatif.

Comment calculer l'activité unitaire?

activité unitaire	= nombre d'occurrences de record membre auxquelles on a accédé ou que l'on a créés et que l'on veut rattacher à une occurrence de record maître*

*: cela est équivalent à la probabilité de rattacher l'occurrence de record membre.
Cette probabilité est = 1 si le set est obligatoire
$\leq$ 1 si le set est facultatif.

ex :



$\sharp X$: 1
$\sharp Y$: 6 → tentative d'accès à 6 occurrences de Y
Y^+ : 3 → seulement 3 accès sont réussis $\Longrightarrow$ création de 3 occurrences de Y
S^+ : 6

S^- : suppression d'une occurrence d'un type de set.

Elle suppose l'accès préalable à l'occurrence du record membre que l'on veut retirer du set ainsi qu'à 1 occurrence du record maître.

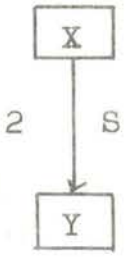
Rmq : la suppression de l'occurrence du record membre n'est pas toujours nécessaire notamment si le type de set est facultatif.

Comment calculer l'activité unitaire?

activité unitaire	= nombre d'occurrences de record membre auxquelles on a accédé ou que l'on vient de supprimer et que l'on veut détacher d'une occurrence de record maître*

*: cela est équivalent à la probabilité de détacher l'occurrence de record-membre.
Cette probabilité est = 1 si le set est obligatoire
≤ 1 si le set est facultatif.

Ex. :



#X : 1
#Y : 2
S⁻ : 2x0,5 = 1
je ne détache qu'une occurrence sur deux

Pour un S.S.L. issu d'un M.E.D. en consultation (lecture), on accède à l'ensemble du S.S.L. en utilisant les primitives suivantes:

- #R
- S
- S^{-1}
- S^F

#R : accès à une occurrence d'un type de record.

Ici, cette primitive ne sert qu'au positionnement sur une occurrence du record 'point d'entrée' du S.S.L.

activité unitaire = 1

S : accès d'une occurrence de record maître à la première occurrence du record membre du set considéré.

activité unitaire = 1x le nombre d'occurrences du record maître

S^F : accès d'une occurrence de record membre à l'occurrence de record membre suivante.

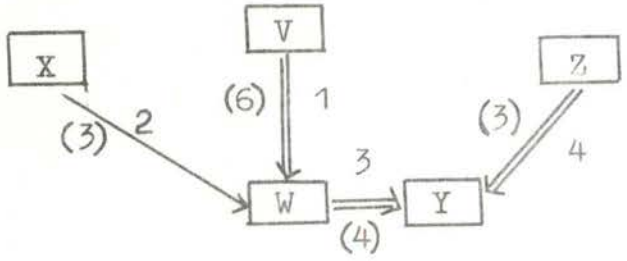
activité unitaire = (cardinalité moyenne du set considéré - 1* x nombre d'occurrences de record maître

*: car on a déjà accédé à la première occurrence par S.

S^{-1}: accès d'une occurrence de record membre à une occurrence de record maître.

activité unitaire = nombre d'occurrences de record membre

Ex. : parcours d'un S.S.L.



#V : 1
S_1 : 1
S_1^F : 1x(6-1)=5
S_2^{-1} : 6
S_3 : 6
S_3^F : (4-1) x [illegible] = 18
S_4^{-1} : 24

• Comment calculer l'activité totale par période pour toutes ces primitives ?

activité totale = activité unitaire × nombre d'activations par période.

Cette démarche étant faite pour chaque sous-schéma logique, il suffit maintenant de cumuler l'ensemble des activités sur le M.L.D.
On disposera alors d'un M.L.D. valorisé, traduction fidèle du M.C.D.

(12.) Optimisation du M.L.D. valorisé

Il convient d'optimiser ce modèle c.-à-d. de chercher parmi des structures équivalentes au M.L.D. obtenu, celle qui minimise une certaine fonction économique qui tienne compte à la fois de l'activité des applications à mettre en oeuvre et du volume des données à traiter.
La fonction économique utilisée est :

F= (nbre accès par période X prix d'accès) + (nbre de caractères stockés X prix du stockage d'un caractère par période).

Une des hypothèses de base pour l'utilisation de cette fonction est que l'activité d'une application peut être mesurée en nombre d'accès quelle que soit la programmation.
On suppose également :

- que le contexte d'utilisation n'est pas pris en compte et que l'on ne considère pas le fait qu'une ressource soit disponible ou non en fonction des requêtes des différents utilisateurs;

- que la mise en oeuvre des structures d'accès nécessaires suivant les différentes techniques provoque une augmentation de volume qui est a priori indépendante de la technique utilisée (les pointeurs sont nécessaires et ont une taille à peu près constante);
- que les machines utilisées ont une structure de mémoire simple: mémoire centrale - mémoire secondaire ; en mémoire centrale, le travail est supposé avoir un coût nul, le transfert d'un bloc d'une mémoire secondaire vers la mémoire centrale coûtant, lui, un accès si la longueur du bloc transféré est inférieure à une certaine taille T max (on s'arrangera pour qu'il en soit toujours ainsi);
- que le prix d'un accès, dans le cas d'un tarif, est fixé quelle que soit la méthode d'accès (en général, il en est ainsi);
- que les volumes stockés proviennent des champs contenus dans les records.

Chacune des primitives sera donc dotée d'un équivalent accès nécessaire à l'utilisation de la fonction économique.

C'est ainsi que l'on aura :

Primitive	équivalent accès	Commentaires
#R	1	Le record est un enregistrement physique accessible par une clé ⟶ le temps d'accès à un record par sa clé est l'unité d'équivalent accès.
R^+	2	Il s'agit d'une recherche d'espace disque et d'une écriture de record.
R^-	1	C'est uniquement une opération de libération d'espace.
RM	1	Le record est déjà lu (#R); il s'agit uniquement d'une réécriture.

$S=S^{-1}=S^{f}$		1		les sets sont réalisés en utilisant une représentation par pointeur physique d'où ces opérations sont équivalentes à l'accès record.
		ordonné	non ordonné	
S^{+}	simple	n/2 + 1	2	Les opérations S^{+} et S^{-} peuvent avoir des valeurs très variables en fonction de l'ordonnancement (ou non) du set et de l'existence d'un chaînage simple ou double (pointeurs avant-arrière). n est ici la cardinalité moyenne du set.
	double	n/2 + 4	4	
S^{-}	simple	n/2 + 1	n/2 + 1	
	double	4	4	

On aura aisément la traduction du M.L.D. valorisé en terme d'équivalents accès, ce qui permettra d'obtenir une valeur économique de référence grâce à la fonction économique et ce, après avoir pris en compte les contraintes de volume. Le concepteur décide alors des modifications qu'il pourrait envisager pour réduire ce coût :

- création de redondances (migrations, créations de nouveaux chemins d'accès, sets redondants,.....)
- abandon de sets ordonnés (qui sont pénalisants pour les mises à jour)
- remise en cause du M.C.D.(exceptionnel car tout est alors à refaire !)
    - redéfinition de certains individus
    - remise en cause des cardinalités.

Il suffit (!!) alors de redéfinir les S.S.L. et de reprendre une nouvelle valorisation du M.L.D.

Grâce à la fonction économique, le concepteur pourra choisir la solution qui réalise le meilleur équilibre entre le nombre d'accès et le volume stocké.

Il continuera ses tentatives d'amélioration du M.L.D. jusqu'à ce que les gains sur l'activité deviennent inférieurs à un certain seuil (par ex. : 10 %).

Rmq. : il va de soi que cette optimisation n'est possible que si l'on dispose d'outils informatiques capables de prendre en charge l'élaboration des S.S.L. et la valorisation de ces derniers.

IV 13. Passage au niveau physique (p.m.).

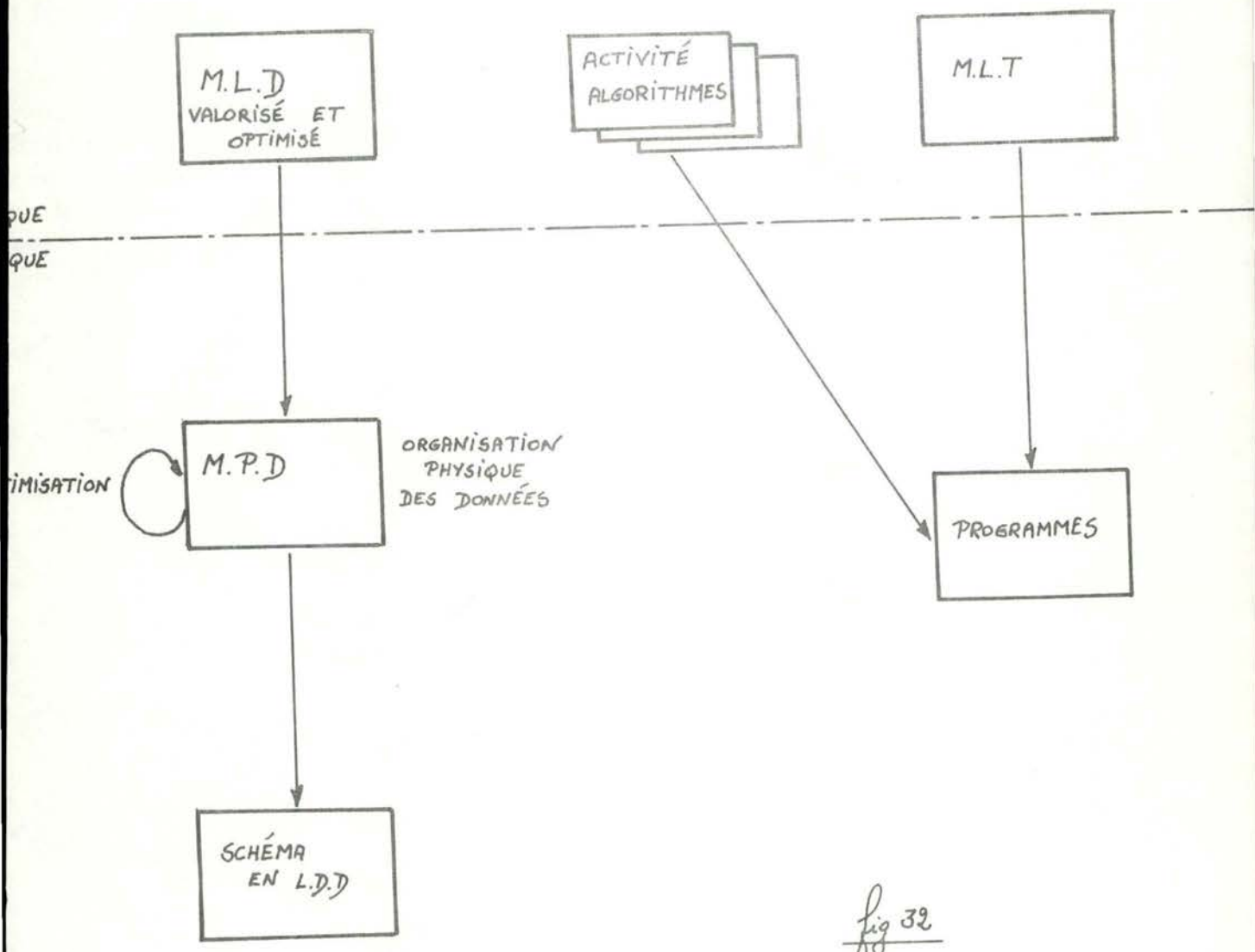


fig 32

IV. 2. Méthode 'NAMUR' : les étapes et leur articulation. (1)

IV. 21. Description conceptuelle du S.I.

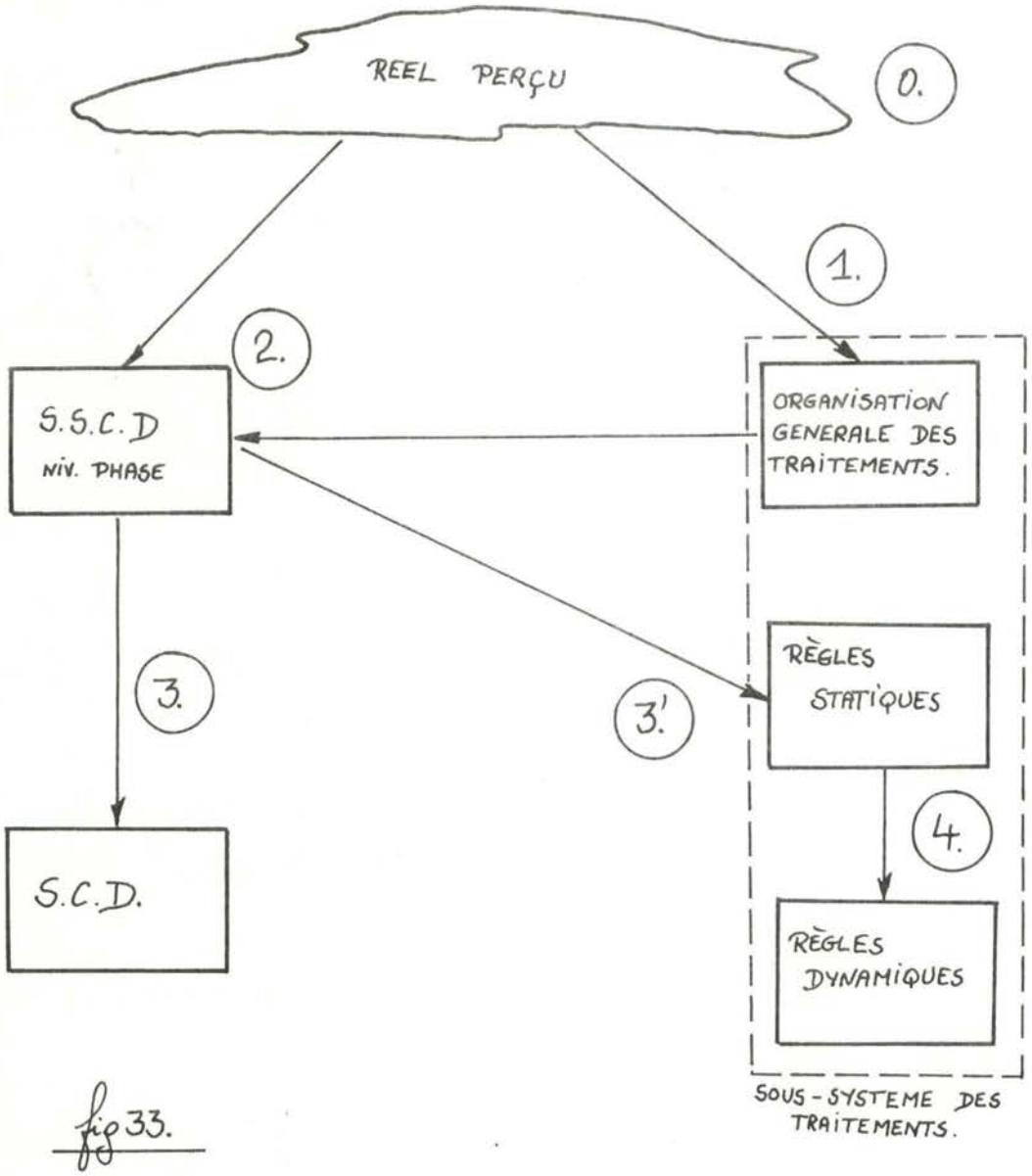


fig 33.

(0.) Constitution du réel perçu.

Le R.P., comme son nom l'indique, constitue une perception du réel propre à l'organisation. C'est une image plus ou moins précise et complète de cette organisation et de son activité. Il comprend l'ensemble des informations qui ont un sens pour elle et qui seront nécessaires à l'élaboration de son S.I. . (cfr. chap IV. 11. 1)

---

(1) 'Modèles et outils d'aide à l'analyse des spécifications fonctionnelles d'un S.I.' F. BODART et Y. PIGNEUR .

1. Identification et structuration générale des traitements .

A partir du R.P., on dégage les différents types de traitement en fonction des critères d'individualisation propres à chacun d'eux. (cfr. chap II. 1.2. p et suivantes)
C'est ainsi que l'on s'attache à reconnaître les différentes phases et applications (les secondes n'étant, comme nous l'avons vu, que des enchaînements de phases relatives à un même flux d'informations) ainsi qu'à détailler les phases en fonctions.

2. Elaboration des sous-schémas conceptuels de données (S.S.C.D.).

En utilisant le modèle entité-association, il faut maintenant définir les S.S.C.D. qui correspondent principalement aux phases que nous avons isolées lors de l'étape 1 . On utilise pour cela des informations du réel perçu (pas toutes car on a fait un certain nombre de choix et d'hypothèses par rapport au R.P. : ex. : pour les entités; on décide de les représenter par un ensemble de propriétés, mais pas toutes celles qui pourraient être contenues dans le R.P.) .On définit et on détaille ainsi les entités et les relations, les connectivités des relations, les différentes contraintes d'intégrité,...
Les S.S.C.D. (ainsi que le schéma conceptuel plus tard) doivent être mis sous forme canonique (dans un but de classification et de clarté mais sans "arrière pensée" de niveau logique) c.-à-d. qu'ils doivent respecter une contrainte fondamentale : la NON REDONDANCE : élimination - des synonymes,
- des homonymes (polysèmes),
- des propriétés redondantes (calculables),
- des associations redondantes (dérivables).

Les S.S.C.D. doivent aussi être validés :

- formellement, syntaxiquement, par rapport à un ensemble de règles de construction (ex : pas d'entités sans propriétés, pas d'associations qui relieraient des entités non définies, ...)
- par les différentes classes d'utilisateurs (validation sémantique) .

3. Construction du schéma conceptuel des traitements (S.C.D.).

Le S.C.D., schéma associé au sous-système des traitements obtenu par ailleurs, se construit assez facilement par intégration des différents S.S.C.D. définis lors de l'étape 2 . On peut procéder de la façon suivante :

- tout S.S.C.D. est sous forme canonique,
- on prend alors deux S.S.C.D., on les regroupe et on tente :
  - . de détecter les synonymes et les homonymes au niveau des propriétés,
  - . de détecter les entités homonymiques,
  - . de détecter les entités synonymiques et éventuellement les intégrer,
  - . de détecter les associations homonymiques et synonymiques,
  - .. ...

  de telle facon que l'on obtienne un nouveau S.S.C.D. sous forme canonique,
- on recommence l'opération jusqu'à ce qu'il ne reste plus qu'un S.S.C.D. c.-à-d. le S.C.D. lui-même, ce dernier étant aussi sous forme canonique.

Le S.C.D. devra, lui aussi, faire l'objet de validations.

3' Description statique des traitements.

Il s'agit de spécifier maintenant les éléments (MESSAGES et structures de données) en entrée et en sortie des différents types de traitements (cfr. fig.12 chap. II. 1.2.).

C'est également à cette étape-ci que l'on spécifie les règles de traitement associées aux fonctions. Ces règles devront être vérifiées c:-à-d. que l'on devra s'assurer qu' :

- elles ont en entrée un certain nombre de propriétés,
- elles ont des propriétés en sortie,
- les propriétés en sortie sont le résultat de l'application des règles aux propriétés d'entrée,
- les propriétés en entrée et en sortie de la règle appartiennent au S.S.C.D. associé à la phase dont la fonction considérée fait partie .

Rmq : cette étape peut être effectuée avant l'étape 3.

(4.) Description de la dynamique des traitements.

On ajoute les spécifications dynamiques à la structure de traitements obtenue à la fin de l'étape 1 et ce, afin de regrouper et de sychroniser :

- les différentes fonctions dans les phases,
- les différentes phases dans les applications.

Ces spécifications dynamiques utilisent les concepts déjà décrits de "point de synchronisation", "événement" et "processus (cfr. chap. II.1.2.).

On obtient ainsi le sous-système des traitements (types de traitements + description statique + description dynamique) .

On dispose maintenant d'une représentation en principe correcte de l'activité de l'organisation

IV.2.2. Description de niveau logique du S.I.

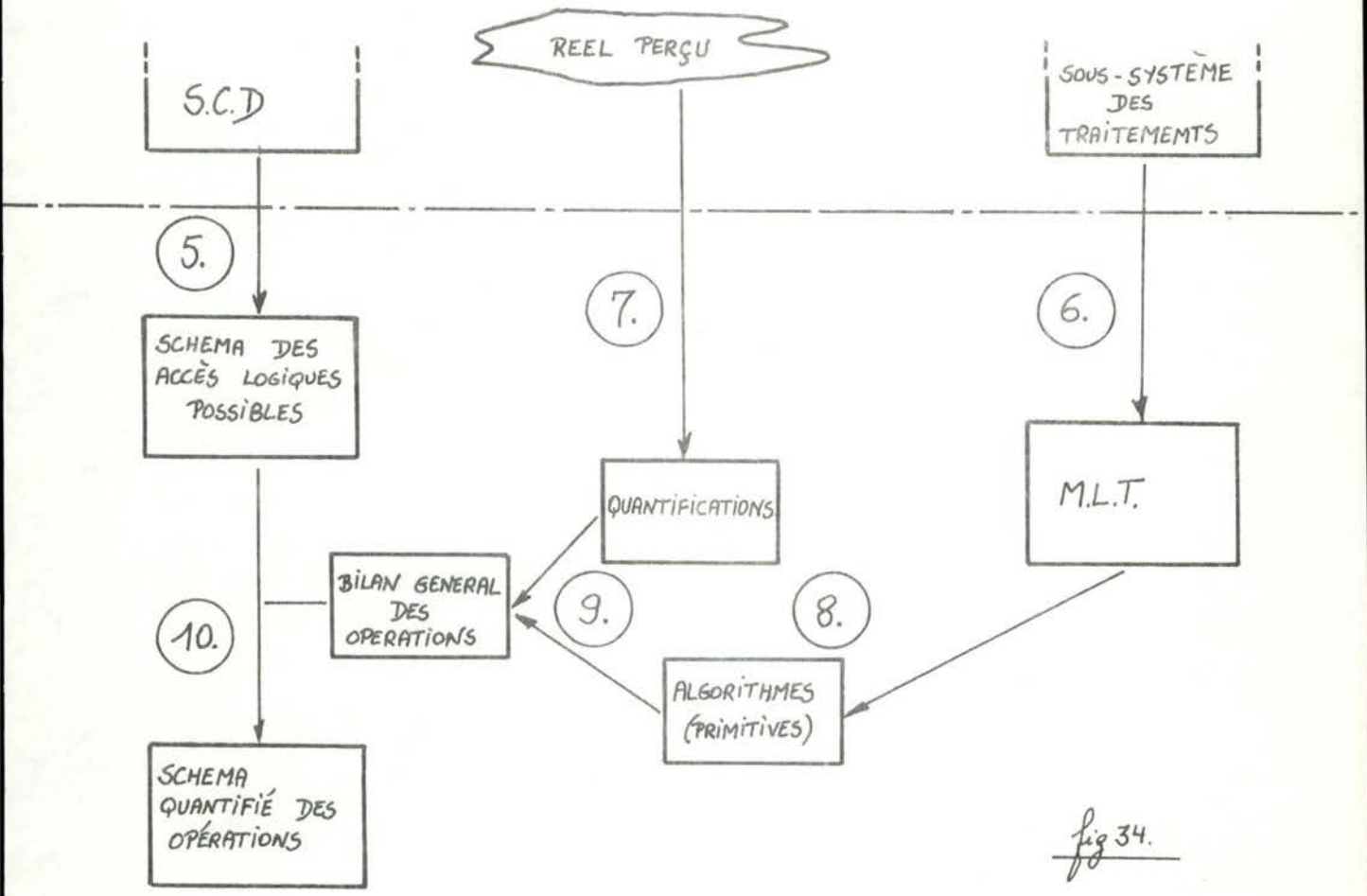


fig 34.

(5.) MAPPING S.C.D. → Schéma des accès logiques possibles.

Ce MAPPING a déjà été étudié au chapitre III (p.96).

(6.) Passage du sous-système des traitements au modèle logique des traitements.

Ce point a également été évoqué au chap. III (p.98).
Rappelons simplement qu'il consiste à reprendre le sous-système des traitements (descriptions statique et dynamique comprises) de niveau conceptuel et à y inclure toutes les considérations d'ordre organisationnel contenues dans le R.P. On établira donc ici l'architecture des moyens de réalisation en décrivant les processeurs et les ressources qui seront utilisées.

7. Prise en compte des quantifications.

Il faut maintenant dégager du R.P. un certain nombre de quantifications qui vont nous permettre d'évaluer l'activité sur le schéma des accès possibles.

On isolera ainsi un ensemble de nombres significatifs (1) tels que :

- le nombre d'entités de chaque type (ex. : 40.000 CLIENTS),
- le nombre moyen d' occurrences de chaque relation par élément origine (ex. : une commande a, en moyenne, 5 lignes),
- une classification des entités de certains types (ex. : 350 commandes : 315 commandes correctes 35 cas litigieux),
- une estimation de délais,
- ...

Ces quantifications vont déjà servir à l'étape suivante lorsqu'il faudra dresser le bilan de l'activité des algorithmes sur les données.

8. Construction d'algorithmes.

Pour chaque traitement de type 'fonction'du M.L.T., on va spécifier un ou plusieurs algorithmes. Ceux-ci se basent sur l'utilisation de primitives : actions élémentaires sur une structure de donnée. Citons, à titre d'exemple, les primitives (2) :

- d'accès à un article en fonction d'une valeur d'item :
  ACCES article (item)
- d'accès à un article à partir d'un autre article :
  ACCES article (article)
- de création (ou de suppression) d'un article :
  CREER (SUPPRIMER) un article
- de modification d'un article :
  MAJ article (item)

---

(1) certains calculs préliminaires sont parfois nécessaires afin d'obtenir ces chiffres.

(2) J.L. HAINAUT propose un ensemble de primitives propres aux modèles des accès logiques que nous avons vu dans 'Un modèle de description de fichiers : le modèle d'accès'.

Avec ce petit jeu de primitives, on peut déjà construire l'algorithme (très simplifié !) de mise à jour des stocks lors de l'enregistrement d'une commande :

APPLICATION : .....

PHASE : .....

FONCTION : mise à jour des stocks lors de l'enregistrement d'une commande.

```
Pour un C = COM-CLI
   Pour chaque L = LIGNE-CDE de C
      Pour P = PRODUIT de L
         SI I-EPUIS (P) = "E"
         Alors MAJ L (Q-DUE-C); marquer L comme "épuisé"
         Sinon MAJ L (Q-DUE-C)
               MAJ P (Q-EN-STOCK, Q-DIFFEREE)
         Fin SI
      Fin P
   Fin L
  MAJ C (ETAT-CC)
Fin C
```

On construit de la même façon un ou plusieurs algorithmes pour toutes les fonctions que l'on peut trouver dans le M.L.T.
De plus, chaque fois qu'un algorithme est terminé, on peut prendre en compte les quantifications qui le concernent et dresser son bilan des opérations.
Ainsi, pour l'algorithme ci-dessus, on a :

- quantifications :
  - 340 commandes vérifiées sont traitées chaque jour ,
  - une commande a en moyenne 3 lignes,
  - la probabilité qu'une ligne de commande demande un produit épuisé est : 0,02
- algorithme transformé :

```
340 X { COM-CLI courante :
          3 X { accès LIGNE-CDE de C;
                accès PRODUIT de L;
                   0,02 X { MAJ L (Q-DUE-C) };
                   0,98 X { MAJ L (Q-DUE-C);
                            MAJ P (Q-EN-STOCK, Q-DIFFEREE) } };
          MAJ C (ETAT-CDE) }
```

bilan des opérations :

type	par cde	par jour
accès COM-CLI courante	1	340
accès LIGNE-CDE (C)	3	1020
accès PRODUIT (L)	3	1020
MAJ L (Q-DUE-C)	3	1020
MAJ P (Q-EN-STOCK, Q-DIFFEREE)	3	1020
MAJ C (ETAT-CDE)	1	340

(9.) Bilan général des opérations

On reprend maintenant tous les bilans des opérations que l'on a construits pour chaque algorithme et on dresse un bilan général des opérations.

On a donc ainsi :-la liste des primitives utilisées pour l'ensemble des algorithmes,

-pour chaque primitive : le nombre de ses activations

(10.) Passage au schéma quantifié des opérations

Cette dernière étape consiste à reprendre le schéma des accès logiques possibles, de le confronter avec le bilan général des opérations afin de voir quels sont les accès réellement utilisés et enfin, de porter sur le schéma obtenu, les résultats du bilan général des opérations.

On obtient ainsi le schéma quantifié des opérations, schéma qui servira de base à l'implantation de la future B.D.

Rmq : on pourrait aussi représenter le passage au niveau logique de la description du S.I. de la façon suivante :

(1.) MAPPING S.C.D. (S.S.C.D.) ⟶ S.A.L.P. (S.S.A.L.P.)

(2.) Passage S.S.T. ⟶ M.L.T.

(3.) Prise en compte des quantifications.

4. Construction d'algorithmes qui travaillent sur les S.S.A.L.P.

5. Bilan des opérations pour chaque algorithme.

6. Elaboration des S.S.Q.O.

Pour chaque algorithme, on reprend le S.S.A.L.P. associé et on le confronte avec son bilan des opérations; on obtient ainsi le sous-schéma quantifié des opérations.

7. Elaboration du S.Q.O.

Le schéma quantifié des opérations s'obtient alors très facilement en intégrant les différents S.S.Q.O.

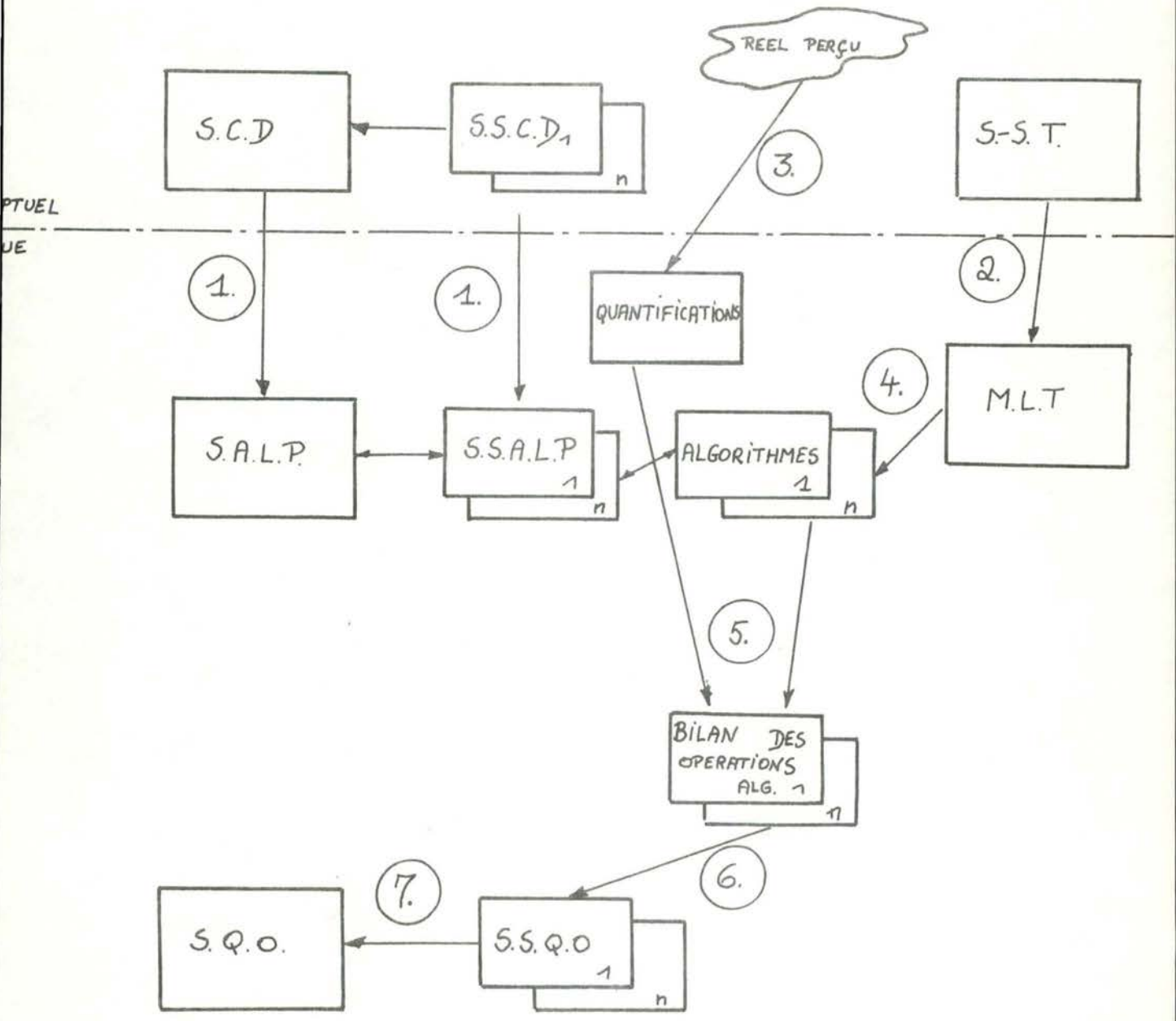


fig 35.

Rmq : A la fin de la description de niveau logique du S.I., il reste à EVALUER :

- l'utilisation des ressources :
  - il s'agit d'évaluer la faisabilité des spécifications conceptuelles compte tenu de la disponibilité d'un ensemble de ressources et de moyens (niveau logique),
  - rappelons que l'utilisation des ressources "data" a été considérée lors du bilan général des opérations (étape 9);
- l'efficacité des algorithmes : on envisage à ce propos de développér un générateur automatique d'une "maquette" de programme ('throw-away' code) à partir de la spécification de l'algorithme et ce, afin d'évaluer le caractère effectif des règles de traitement et de l'algorithme .

IV 2.3 Passage au niveau physique de la construction du S.I. (p. m).

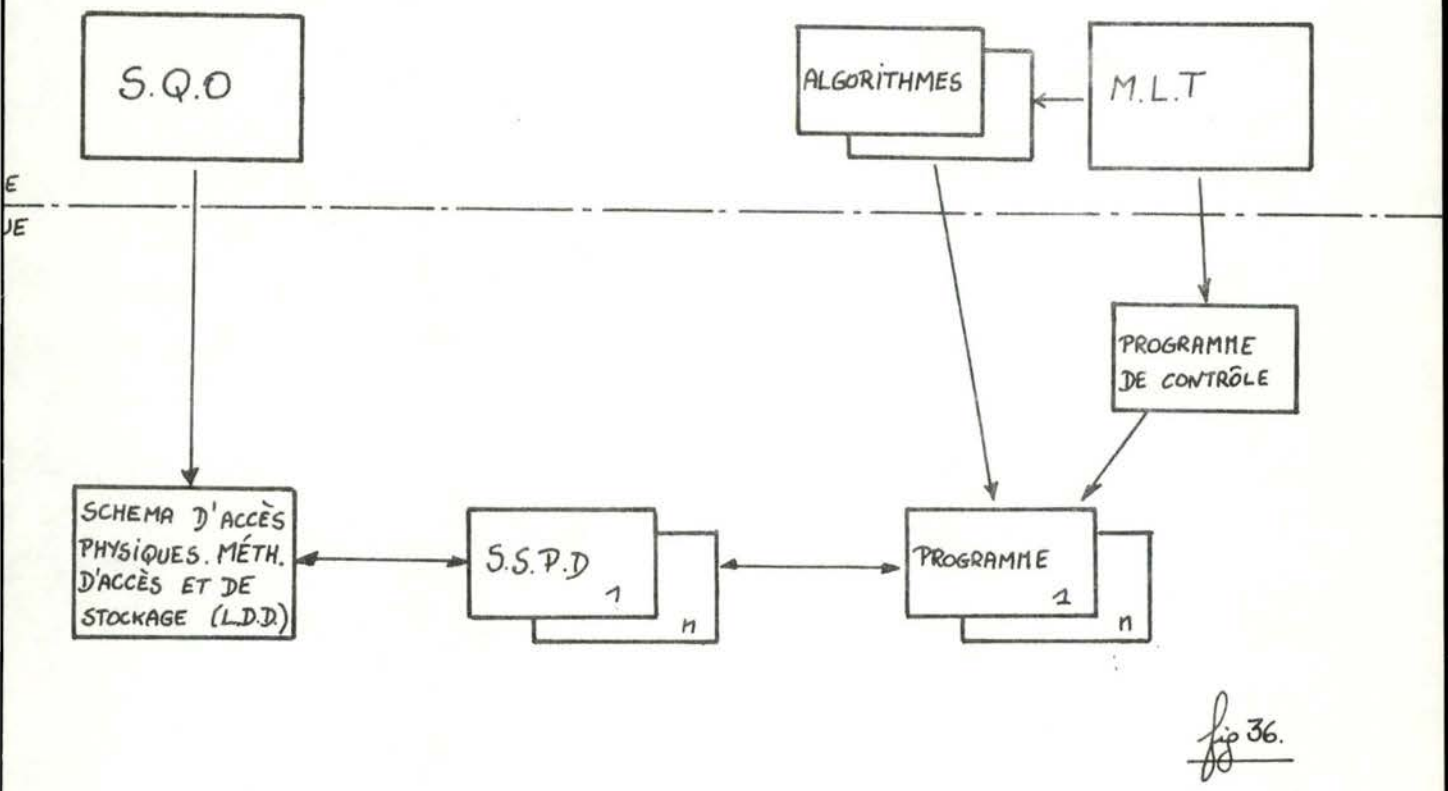


fig 36.

IV 3 Commentaires.

Nous venons donc de voir les méthodes proposées par MERISE et 'NAMUR'; il nous reste maintenant à les comparer.

Niveau conceptuel de la description du S.I.

1) MERISE s'attache directement à élaborer un M.C.B.S. à partir du réel perçu et ce n'est qu'après qu'elle fait intervenir les différentes vues particulières (modèles externes) pour valider le M.C.B.S. proprement dit.

Pour 'NAMUR', ces modèles externes (S.S.C.D. – applications, phases) n'ont pas ce rôle de validation puisqu'ils servent à la construction même des schémas de niveau conceptuel. Certains pourraient prétendre qu'en agissant ainsi, on ne tient pas compte de tous les éléments du réel perçu, notamment ceux qui ne servent pas encore aux différents utilisateurs. C'est vrai mais il faut faire un choix : ou on modélise en tenant compte d'éléments qui ne sont pas directement nécessaires aux différentes classes d'utilisateurs (modèles externes) de l'organisation - mais alors quand faut-il s'arrêter? - ou on se limite dans la description du S.I., à une

intégration de modèles externes, comme c'est le cas pour 'NAMUR'. (ici donc, l'application doit être bien définie et on tend alors à construire une B.D. économique).

2) MERISE élabore à l'étape 1 un réel perçu machinable (R.P.M.), sorte de dictionnaire de données qui, comme son nom l'indique, reprend toutes les informations du R.P. et sert de base à de nombreux outils automatiques qui vont aider le concepteur dans sa tâche (cfr. chap.V).

Niveau logique de la description du S.I.

A ce niveau, les deux approches sont tout-à-fait semblables. Elles veulent, en effet, valoriser (ou quantifier) un modèle logique de données (ou un schéma des accès possibles) en tenant compte d'un certain nombre de quantifications et de l'activité des algorithmes sur la structure logique des données. Le calcul de l'activité se fait d'ailleurs de la même façon : on dénombre les primitives utilisées et on considère le nombre de leurs activations durant une certaine période. Il subsiste cependant quelques petites différences. En effet, si MERISE fournit un M.L.D. valorisé unique, la deuxième méthode ('NAMUR') peut donner naissance à plusieurs schémas quantifiés des opérations. Cela provient du fait que l'on peut trouver plusieurs algorithmes possibles par fonction et, qu'a priori, on n'en pénalise aucun à ce niveau. Il peut donc y avoir autant de schémas que d'ensembles différents d'algorithmes. Une deuxième différence, plus importante celle-ci, c'est que MERISE propose une étape d'optimisation au niveau logique en utilisant une fonction économique et en partant du principe que certaines améliorations du M.L.D. sont possibles indépendamment de la réalisation technique et de l'utilisation du S.G.B.D. (pour les types d'amélioration possibles cfr. p.)

Il faut signaler que cette optimisation du M.L.D. ne garantit pas du tout un modèle physique des données (M.P.D.) optimal; une deuxiéme optimisation sera en effet nécessaire au niveau physique (cfr. fig.32). Certaines contraintes de S.G.B.D. pourraient même réduire à néant l'effet de l'une ou l'autre amélioration apportée au M.L.D. (ex. : migrations de champs ⟶ limitation de la taille des articles.) ou s'avérer être des actions redondantes avec des opérations d'opti-

misation du niveau logique (ex.: limitation des accès ⟶ contraction de sets - regroupement des occurrences du record membre dans le même espace physique que l'occurrence correspondante du record maître (l'utilisation du set a alors un coût nul) -).
Pourquoi alors ne pas attendre le niveau physique pour faire toutes les optimisations?
'NAMUR', en tout cas, opte pour cette solution et n'évaluera son schéma quantifié des opérations qu'au niveau physique lorsqu'on aura fait les mesures de volumes et de performances nécessaires.
Si cette évaluation n'est pas satisfaisante, on peut refaire l'ensemble des transformations pour un autre schéma quantifié des opérations, résultat d'un jeu différent d'algorithmes.
Cela peut sembler assez fastidieux à première vue, mais il faut faire remarquer que ces opérations peuvent largement être réduites en fonction de l'expérience du concepteur de la B.D. qui peut, dès le départ, faire le bon choix.

Chap.V Outils développés dans le cadre des deux méthodes.

Nous venons donc de voir, aux chapitres précédents, comment on peut construire un système d'informations. Nous avons présenté à cet effet deux méthodes, leurs étapes et leur articulation, ainsi que les modèles sur lesquels elles s'appuient. Cependant, on s'aperçoit rapidemment que lorsque l'on veut concevoir et réaliser un système d'informations un peu complexe, on se trouve devant une tâche qui devient très vite fastidieuse et compliquée (documentation énorme, nombreuses vérifications, mappings à faire manuellement,...), ce qui est évidemment source d'incohérences et d'erreurs. C'est pourquoi, pour faciliter le travail du (ou des) concepteur(s), on a développé des outils informatiques d'aide à la conception et à la réalisation de systèmes d'informations. Ce sont ces outils que nous allons présenter brièvement dans ce chapitre tout en rappelant que pour certaines étapes de la méthode, une automatisation paraît difficile voire impossible dans la mesure ou les choix à faire sont assez complexes.

V.1. Outils proposés par MERISE.

MERISE propose de mémoriser les états successifs du S.I. de la conception à la réalisation, dans une B.D. particulière, un dictionnaire de données : CATI (Catalogue Automatisé des Traitements et des Informations).

Elle fournit ensuite un ensemble d'outils spécifiques qui vont travailler sur cette B.D. CATI par l'intermédiaire du S.G.B.D. MIISFIIT développé également à Aix en Provence par l'équipe d'H. TARDIEU : le concepteur n'a plus alors qu'à 'alimenter' les outils en leur fournissant les informations voulues. Mais avant de donner la liste et de décrire ces outils, voyons comment ils s'agencent avec la méthode, les modèles et le CATI. de MERISE.

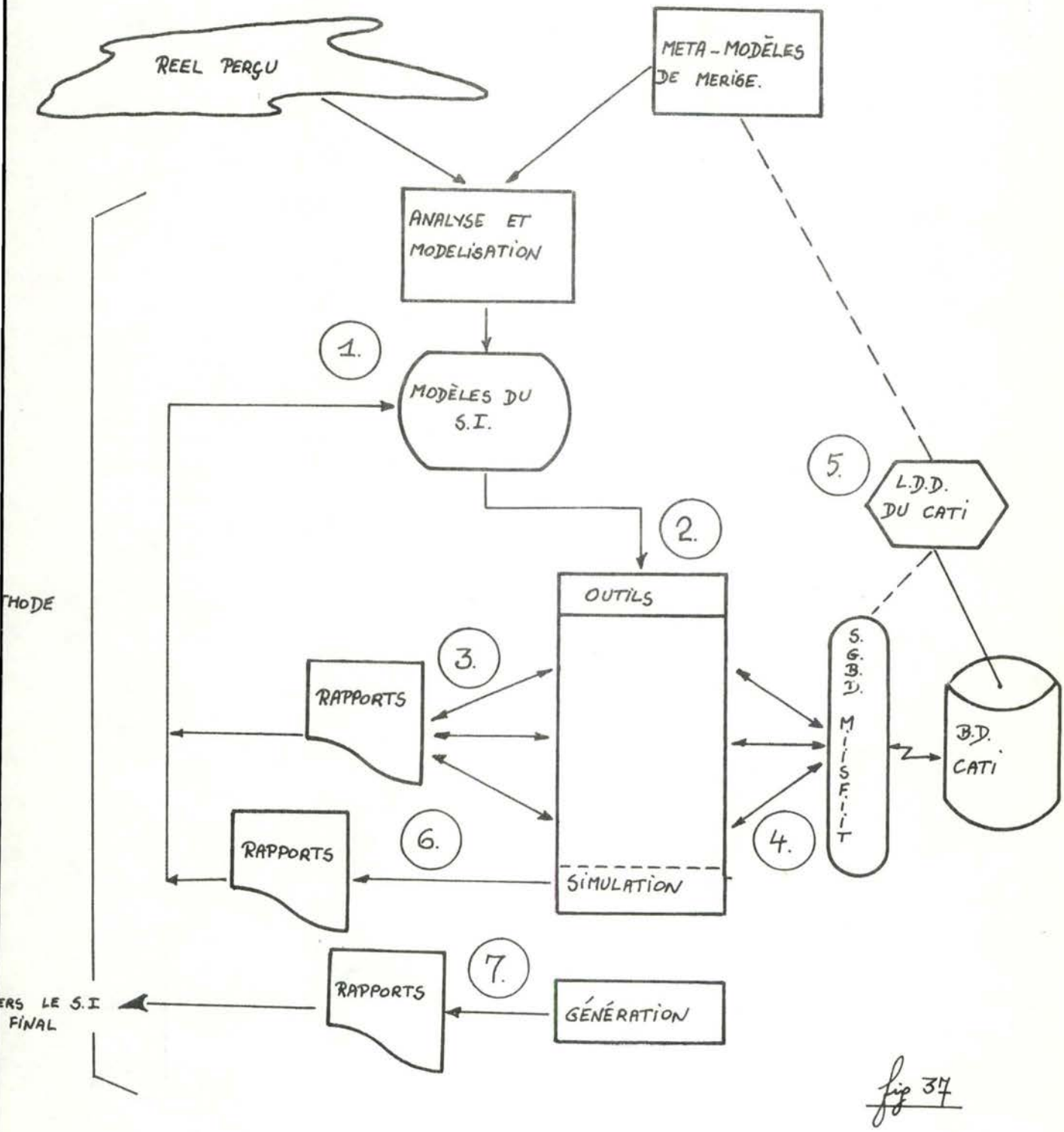


fig 37

(1) A partir du réel perçu, des méta-modèles de MERISE (cfr. chapII) et éventuellement de rapports de documentation (s'il est déjà possible d'en éditer), le concepteur élabore les différents modèles de futur S.I.

(2) Le concepteur est aidé dans sa tâche par un ensemble d'outils qui réalise un certain nombre des étapes de la méthode MERISE et à la réalisation d'autres :
Ce sont par exemple les outils
- d'alimentation du CATI,
- de vérification et de contôle de cohérence,
- de validation,
- de transformation d'un niveau à un autre,
- de génération de rapport de documentation (rapports qui peuvent s'avérer très utiles lors de la conception du S.I.) (3)
- ...

(nous en donnerons une liste précise ci-après.)

(4) Ces outils sont en fait des programmes COBOL ou ASSEMBLER qui utilisent le L.M.D. propre ou S.G.B.D. MIISFIIT pour travailler sur le CATI.

(5) De plus, le lexique ou L.D.D. du CATI doit permettre une description physique des informations propres à chacun des méta-modèles de MERISE mais sous une forme compatible également avec le S.G.B.D. MIISFIIT dont on utilise le L.M.D.

(6) En plus des vérifications que l'on peut qualifier de 'statiques', un contrôle efficace des spécifications dynamiques d'un modèle conceptuel/logique de traitements peut être réalisé par des simulations dont les résultats peuvent permettre de réajuster les spécifications que l'on a effectuées.
Si cet outil n'est pas encore réalisé à l'heure actuelle, des études théoriques ont déjà été faites, notamment au niveau conceptuel des traitements.
Nous allons vous en livrer, à titre indicatif, les résultats.

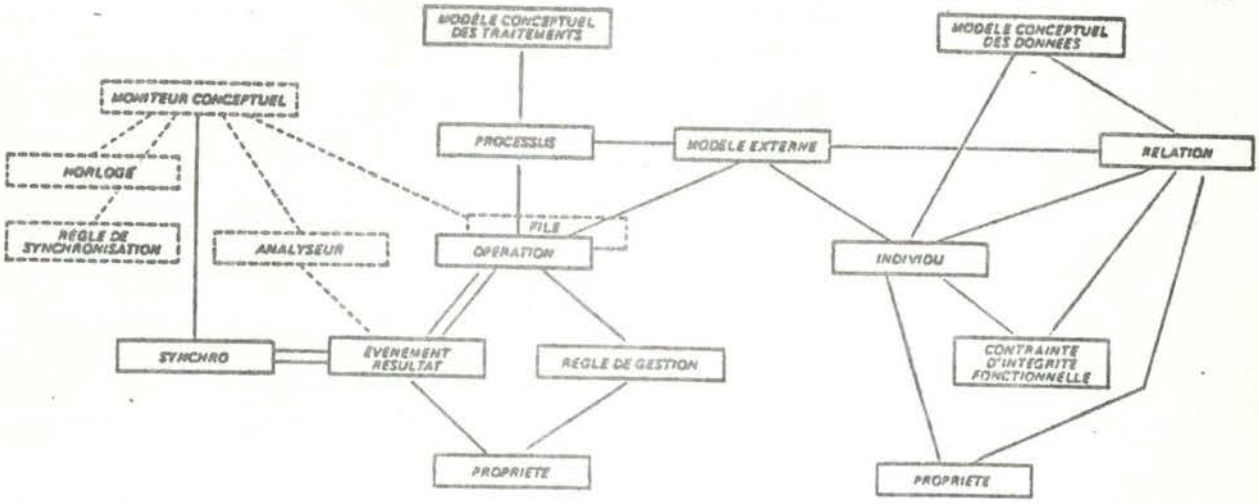
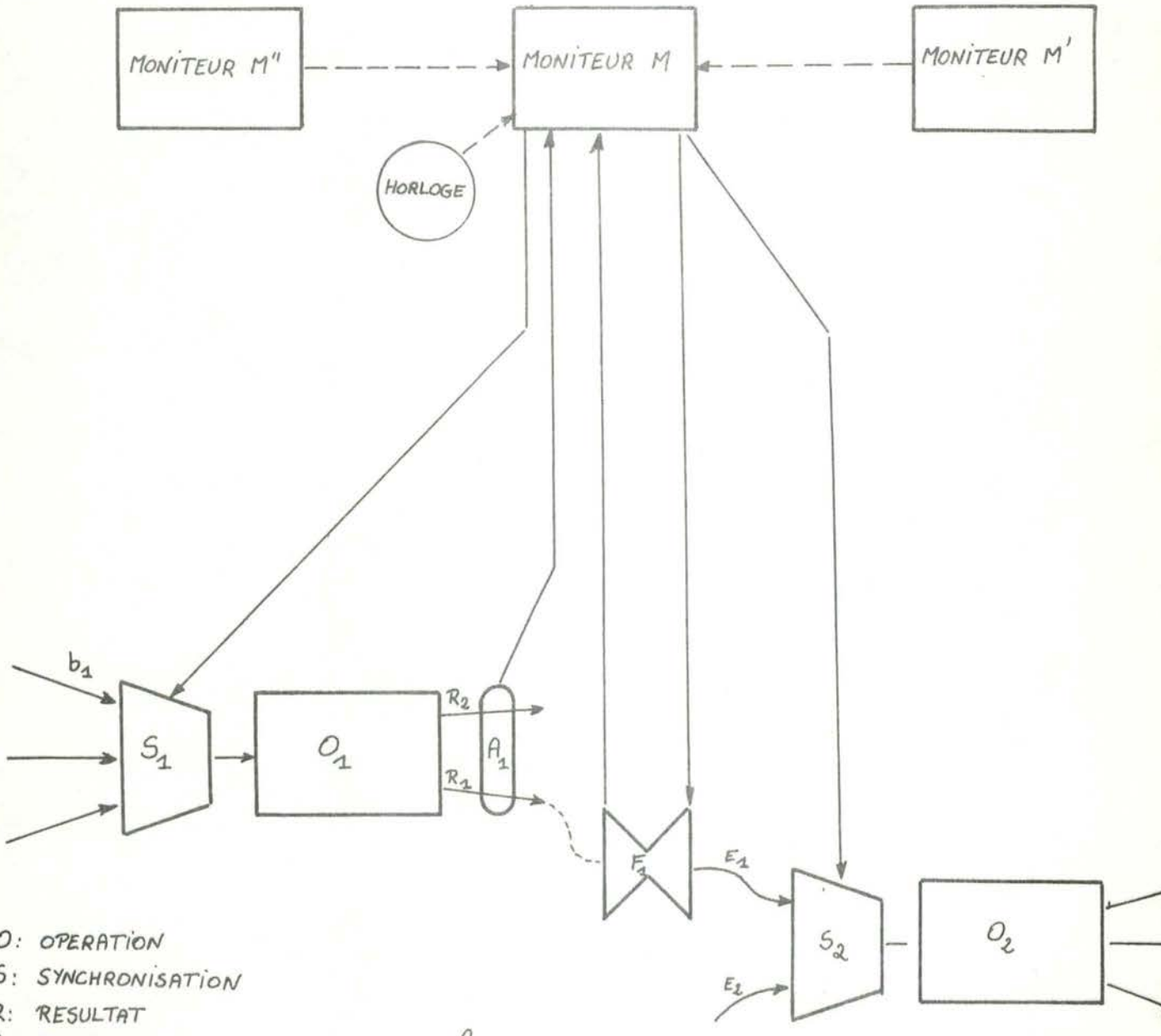


fig 38 : MODÈLE CONCEPTUEL SYSTÈME



O: OPERATION
S: SYNCHRONISATION
R: RESULTAT
A: ANALYSEUR
F: FILE D'ATTENTE
E: EVENEMENT

fig 39

Moniteur conceptuel, horloge, règle de synchronisation, analyseur et file d'attente sont des concepts qui se superposent au M.C.T. et n'en font pas, en tant que tels, partie.
Cependant, en agissant sur les synchronisatons, les événements/résultats et les opérations, ils permettent (ou devraient permettre, car aucune application utilisant ces notions n'est réalisée pour le moment) de contrôler le fonctionnement général du modèle, voire même le simuler.

Fonctionnement général.

Remarques préliminaires.

- A chaque résultat correspond, pour l'analyseur(1), une variable d'état indiquant le fait qu'il a déjà été produit ou non.
- Si un résultat joue dans la suite le rôle d'un événement, il y aura également, pour le moniteut (1), une variable d'état indiquant l'existence de cet événement.

L'analyseur assure les tâches d'analyse et de composition booléenne des variables d'état associées aux résultats de l'opération qui le précède.
Il transmet les résultats sans les modifier.
Lorsque la proposition booléenne correspondante est vérifiée, il délivre une variable d'état qui rend compte de la fin de l'activité de l'opération.
Cette variable d'état peut, en fonctionnement asynchrone, piloter des synchronisations ou des files d'attente (1) tandis qu'en fonctionnement synchrone, elle est utilisée par un moniteur (1) de façon à assurer le synchronisme général.
De plus, en fonctionnement synchrone, l'analyseur positionne les variables d'état des événements associés aux résultats produits.

Exemple: en fonctionnement asynchrone.

---

(1) cfr. ci-après.

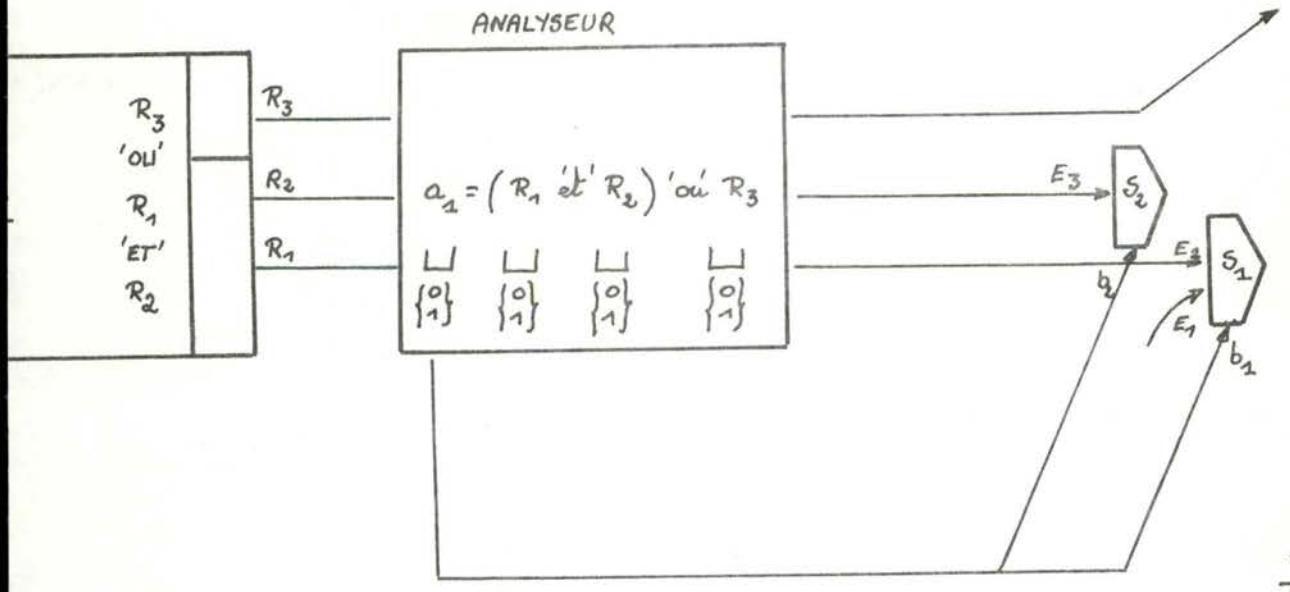


fig 40

L'opération OP_1 émet $\begin{cases} \text{soit } R_1 \text{ et } R_2 \\ \text{soit } R_3 \end{cases}$

Lorsque l'état a_1 est atteint ($a_1 = 1$), les variables b_1 et b_2, souvent appelées 'majeures' de la synchronisation, sont positionnées (b_1 et b_2 contribuent alors aux synchronisations S_1 et S_2) et les variables d'état associées aux résultats, ainsi que la variable a de l'analyseur, sont remises à 0.

en fonctionnement synchrone.

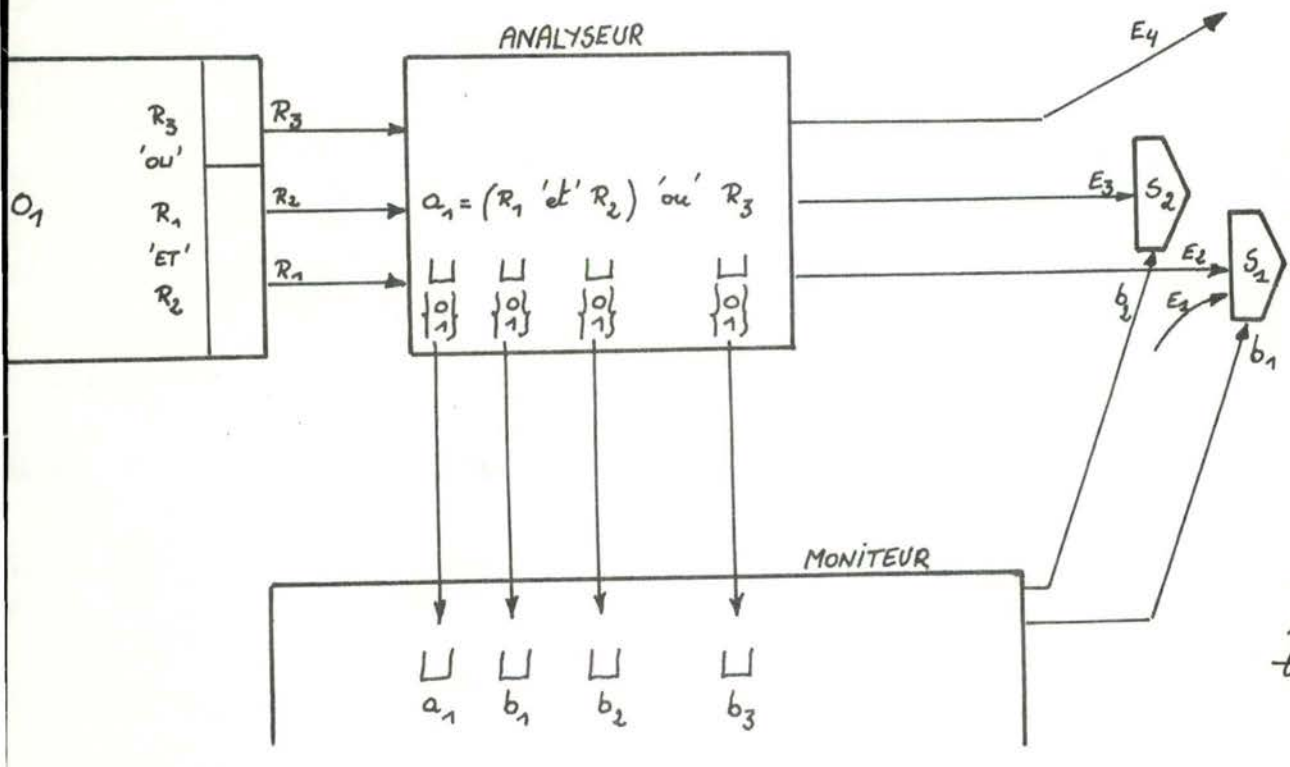


fig 41

- Lorsque l'état a est atteint (a_1= 1), les variables d'état associées aux résultats sont remises à 0.
- Lorsqu'une synchronisation est déclenchée, les variables d'état des événements entrants sont remises à 0 (ces variables sont contenues dans le moniteur).
- Ces variables d'état associées aux événements sont positionnées par les analyseurs en fonction des différents résultats produits.

Cette technique permet le déclenchement de certaines synchronisations même si les opérations ne sont pas entièrement terminées. Il suffit que les événements nécessaires à ce déclenchement soient déjà produits.

Le moniteur conceptuel : ensemble de règles logiques (règles de synchronisation) qui permettent d'assurer la synchronisation générale des opérations d'un ou de plusieurs processus ou du système complet.
Par une analyse et une composition booléenne des variables d'état qu'il reçoit des files d'attente (1) et/ou des analyseurs, il détermine la valeur des variables d'état pilotant des files d'attente et/ou des synchronisations. (cfr. fig.39)
Le moniteur peut être lui-même conditionné par une horloge, un calendrier ou des actions manuelles.
Il peut être également relié à d'autres moniteurs par une ou plusieurs variables d'état (possibilité de répartition ou de hiérarchisation de moniteurs).

La file d'attente : élément permettant de mettre en attente des occurrences d'un type d'événement avant qu'elles ne puissent être traitées par une opération.
Elle est caractérisée par :

- un 'réservoir' de capacité limitée,

- l'émission d'une variable d'état indiquant à tout moment son niveau de remplissage,
- un ensemble de règles permettant de gérer des occurrences en attente et en particulier, de les libérer suivant les règles de la file (LIFO, FIFO, avec ou sans priorité...) et suivant une variable d'état qu'elle reçoit d'un moniteur ou d'un ou plusieurs analyseurs,
- la possibilité d'assurer la distribution des occurrences d'événements aux différentes opérations libres.

Elle permet, en outre, un fonctionnement général partiellement ou totalement asynchrone.

Tout ceci devrait nous permettre de simuler le fonctionnement d'un modèle conceptuel de traitements. Il faut cependant mettre l'accent sur le fait que les moyens de réalisation : processeurs, ressources, n'entrent pas ici en ligne de compte, ce qui enlève une grande partie de l'intérêt d'une telle simulation. On travaille toutefois à Aix-en-Provence sur un outil de simulation qui répondrait à ces exigences.

(7) L'outil de génération devrait fournir, à partir de CATI, un S.I. opérationnel. Il n'en est rien à l'heure actuelle bien que l'on puisse dire que l'on est en mesure de générer les différents modèles de données (M.C.D.-M.L.D.-M.P.D.) obtenus grâce à un ensemble d'outils déjà réalisés. Quant à la partie relative aux traitements, des outils sont en voie de réalisation. Ils devront, eux aussi, faciliter l'élaboration des différents modèles de traitements et permettre, à la fin, leur génération automatique.

Ceci étant dit, voyons quelle est la liste des outils disponibles au 1er janvier 1980.

. Remarque préliminaire :

Tous les outils disposent de leur propre procédure dans CATIPROC. Cette procédure est disponible en donnant la commande : 'CATIPROC (NOM-OUTIL)'.

ALIMCATI

Ce programme permet d'alimenter sur un catalogue 'méthodologie' les informations relatives à un modèle conceptuel; le programme constitue un guide pour cette démarche et effectue de nombreux contrôles.

ALEXCATI

Ce programme permet d'alimenter sur un catalogue 'méthodologie' les informations relatives à un modèle externe relatif à un modèle conceptuel déjà enregistré, contrôlé et décomposé ; ce modèle externe est déclaré soit en mise à jour, soit en consultation.

DES1CATI

Ce programme permet d'extraire les informations nécessaires concernant les individus, relations et contraintes d'un modèle en vue de son dessin automatique sur traceur.

DES2CATI

Ce programme reprend les données extraites par DES1CATI et prépare une bande magnétique utilisable par le traceur.

SYNONYME

Ce programme recherche les synonymes à partir d'une liste de mots-clés fournie par l'utilisateur; sont synonymes pour le programme deux informations qui présentent 3 mots-clés en commun.

LISTMCLE

Ce programme permet d'analyser les mots-clés caractérisant un ensemble d'informations; avec LISTALPHA on obtient la liste alphabétique des mots-clés enregistrés; avec LISTFREQ on obtient leur fréquence d'apparition.

DCMPCATI

Ce programme permet de réaliser la décomposition logique ou physique d'un modèle conceptuel chargé dans le catalogue.

AVALCATI

Ce programme permet de dérouler une arborescence logique sur un grand nombre de niveaux (ex nomenclature, thésaurus) en éditant les informations rencontrées.

AMONCATI

Ce programme permet de remonter une arborescence logique sur un grand nombre de niveaux (ex nomenclature, thésaurus) en éditant les informations rencontrées.

VAMJCATI

Ce programme est destiné à contrôler la compatibilité entre le modèle conceptuel et un modèle externe déclaré en mise à jour.

VACOCATI

Ce programme est destiné à contrôler la compatibilité entre le modèle conceptuel et un modèle externe déclaré en consultation.

COHE

Ce programme permet de contrôler la cohérence interne des cardinalités concernant 3 individus à l'intérieur d'une relation N-AIRE avec N = 3.

METH01

Ce programme permet de calculer la cardinalité résultant de la composition d'une chaîne de relations reliant 2 individus.

INCLUS

Ce programme permet de contrôler la compatibilité des cardinalités entre 2 relations dont l'une est définie comme l'incluante et l'autre comme l'incluse.

CAOMI

Ce programme constitue une aide automatique à la conception d'un modèle conceptuel (ou externe).

EVENCOMP

Ce programme permet de comparer 2 informations en éditant les composants communs à ces informations.

CARDIF

Ce programme permet de contrôler pour les différentes relations d'un modèle conceptuel, la cohérence entre les cardinalités individuelles et les contraintes d'intégrité fonctionnelles.

CHEMIN

A partir d'une liste d'identifiant(s), ce programme recherche s'il existe un ou plusieurs chemin(s) de contraintes d'intégrité fonctionnelle(s), permettant d'atteindre un individu-cible donné.

CYCLE

Ce programme recherche s'il existe, à l'intérieur d'un modèle donné, un ou plusieurs cycles de contraintes d'intégrité fonctionnelles.

CTRLCATI

Ce programme permet de lancer un ensemble de sélections de contrôle sur un catalogue des informations et assure l'édition des erreurs.

ENTICATI

Ce programme est destiné à enrichir un CATI, c'est-à-dire à générer pour tous les composants d'un système NAT INF COM et NOM INF COM lorsque l'on n'a saisi que les NO INF COM; à partir de NO INF COM le programme recherche l'information correspondante et y trouve NAT INF et NOM INF.

CHARCATI

Ce programme permet le chargement d'un CATI en conversationnel Dans une première étape l'utilisateur définit sur quel sous-ensemble il veut travailler; dans une deuxième étape il rentre ses données selon la définition précédente.

FACSIM1

Ce programme permet de charger dans un CATI un ou plusieurs FAC-SIMILES d'état ordinateur; à partir d'une demande de l'utilisateur exprimée dans une carte paramètre le programme sélectionne sur une bande spool le ou les enregistrements requis et les stocke dans le CATI sous forme de ligne de remarque attachée à une information. Ce programme est normalement utilisé en différé.

FACSIM2

Ce programme permet la restitution d'un FAC-SIMILE sur imprimante ou sur terminal, à partir des informations mémorisées dans le CATI.

INFVAL

Cet outil permet l'édition standard des informations d'un CATI avec les valeurs qu'elles peuvent prendre. C'est un état spécial MIISFIIT que l'on peut utiliser en conversationnel (MITSO) ou en simulation (MISSIMUL).

INFLIG

Cet outil permet l'édition standard des informations d'un CATI avec les remarques et les lignes qui les concernent (état spécial MIISFIIT).

EDITCATI

Cette procédure permet de charger sur un CATI nouveau, pourvu d'une bibliothèque MIISFIIT (CATIBIB), les états spéciaux INFVAL et INFLIG.

VIDCATI

Ce programme permet d'extraire les données d'un CATI ou un sous-ensemble sous un format séquentiel (vidage).

PREALIM

Ce programme prépare un CATI strict pour l'utilisation des outils méthodologiques et d'abord l'outil ALIMCATI.

EDIMOD

Ce programme permet d'éditer un modèle conceptuel des données sous une forme normalisée dans le formalisme individuel.

POLYSEME

Ce programme permet de détecter les informations qui possèdent :

- soit le même NOM INF,
- soit le même NO INF.

On rappelle que dans le CATI on dispose de deux identifiants alternatifs :

- NO INF
- (NAT INF+NOM INF).

COHECATI

Ce programme permet de contrôler la cohérence des informations saisies dans un catalogue des informations; une information citée dans un résultat doit figurer sur un événement ou être déductible des données saisies sur un événement à travers une suite de règles de gestion; une information en entrée doit normalement être utilisée soit dans un résultat soit dans une règle de gestion.

RNUMCATI

Ce programme permet de modifier si nécessaire la numérotation des données d'un CATI avant de le fusionner avec d'autres données dans un nouveau CATI. On précède le numéro initial par une lettre caractéristique du système.

CONCATI

Ce programme permet de recharger dans un CATI des données conformes au format de sortie du programme précédent (concatenation).

ALIMVME

Ce programme permet de préciser les renseignements concernant les prévisions de volume relatives à un modèle conceptuel des données déjà mémorisé :

- nombre d'occurrences des individus
- format des différentes propriétés
- cardinalités maximales et moyennes des relations.

Ces éléments sont nécessaires pour les optimisations.

CONCMIL

Ce programme permet de transformer un modèle conceptuel des données en un modèle (interne) logique des données non optimisé.

EDIMLD

Ce programme permet d'éditer un modèle (interne) logique des données sous une forme normalisée dans un formalisme du type CODASYL simplifié.

OPTPHY

Tenant compte de la taille des records et des cardinalités des sets et d'autre part de la taille maximale d'un enregistrement physique (TMAX), ce programme effectue une optimisation du modèle physique des données au moyen de divers regroupements.

MODIMLD

Ce programme permet de nodifier les records et les sets d'un modèle logique des données.

V. 2 Outils proposés par 'NAMUR'.

L'outil principal développé à NAMUR est sans aucun doute D.S.L. (Dynamic Specification Language) (1). C'est un langage développé dans le cadre du projet ISDOS (vaste projet de l'université de Michigan pour la création d'un outil d'aide à l'analyse et au développement d'un S.I.) qui a pour but :

- de décrire les spécifications fonctionnelles d'un S.I. : spécifications de niveau conceptuel (cfr. chap.II 1.2.) et prise en compte d'un ensemble de ressources et de moyens (niveau logique, chap.II.2.2.p.)
- de permettre cette description en considérant aussi les aspects dynamiques d'un S.I. (traitements),
- de mémoriser toutes les spécifications dans une B.D. que l'on pourra exploiter dans la suite.

Schéma global de l'utilisation de D.S.L. (2)

.../...

(1) 'Dynamic Specification Language and a Simulation Model Analyzer for an Information System' NAMUR Version 01 Users Manual F. BODART et Y. PIGNEUR

(2) "Modèles et outils d'aide à l'analyse des spécifications fonctionnelles d'un système d'information." F. BODART et Y. PIGNEUR

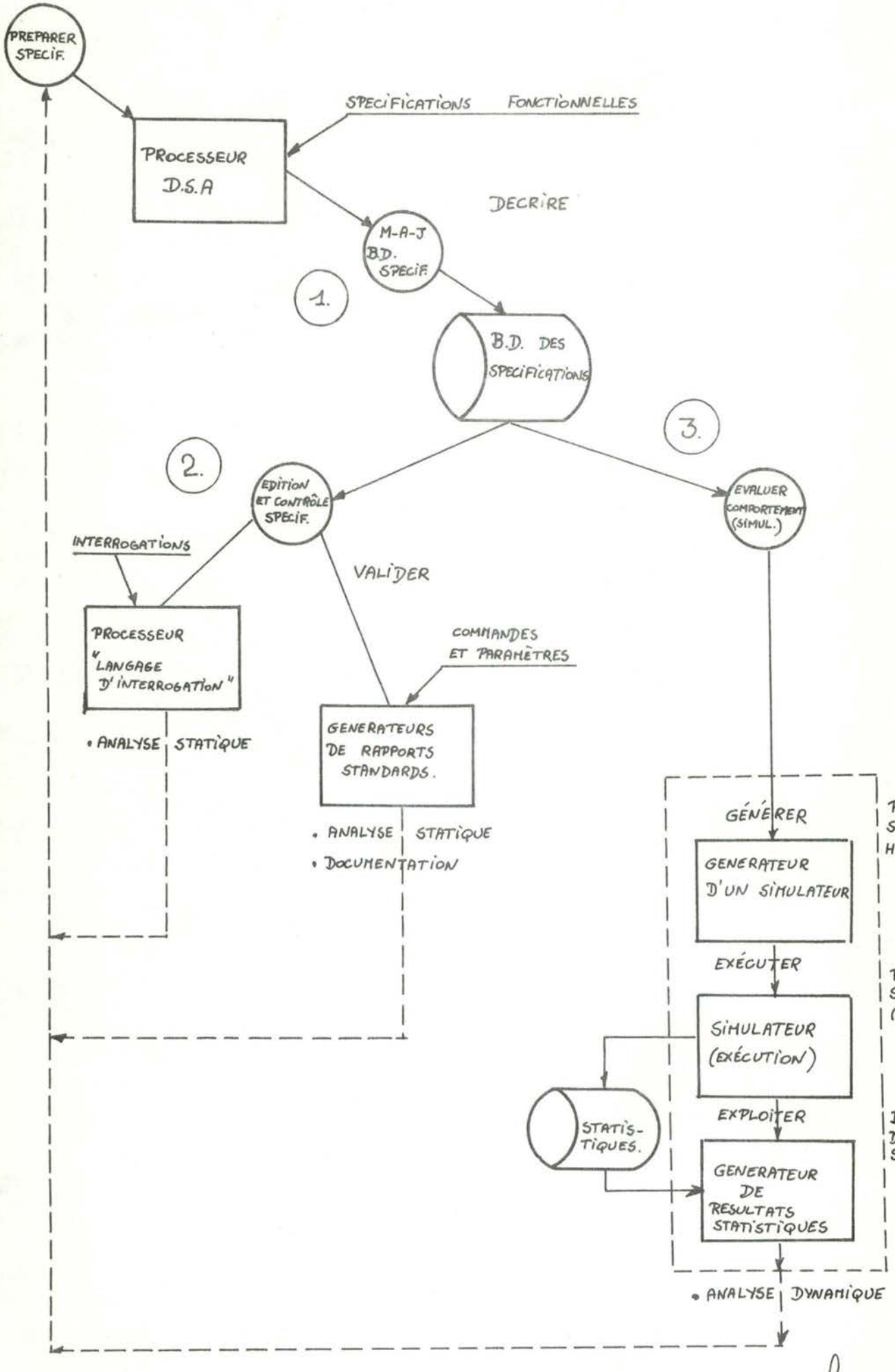

PREPARER SPECIF.
SPECIFICATIONS FONCTIONNELLES
PROCESSEUR D.S.A
DECRIRE
M-A-J B.D. SPECIF.
1.
B.D. DES SPECIFICATIONS
2.
3.
EDITION ET CONTRÔLE SPECIF.
EVALUER COMPORTEMENT (SIMUL.)
INTERROGATIONS
VALIDER
PROCESSEUR "LANGAGE D'INTERROGATION"
COMMANDES ET PARAMÈTRES
• ANALYSE STATIQUE
GENERATEURS DE RAPPORTS STANDARDS.
• ANALYSE STATIQUE
• DOCUMENTATION
GÉNÉRER
GENERATEUR D'UN SIMULATEUR
P.M. DE LA SIMULATION HYP. A. TEST
EXÉCUTER
SIMULATEUR (EXÉCUTION)
P.M. DE LA SIMULATION (EXECUTION)
STATIS-TIQUES.
EXPLOITER
DEMANDES DE RESULTAT STATISTIQUE
GENERATEUR DE RESULTATS STATISTIQUES
• ANALYSE DYNAMIQUE

fig 42.

(1) Les concepteurs du futur S.I. préparent les spécifications fonctionnelles (cfr. chap.I.23.) relatives à ce système, les transforment en spécifications D.S.L., et les entrent ensuite, par l'intermédiaire du processeur D.S.A. (Dynamic Specification Analyzer), dans la B.D. des spécifications. Deux types d'utilisations de la B.D. sont alors possibles.

(2) L'analyse 'statique' des spécifications et de l'édition de rapports de documentation.

Les concepteurs disposent :

- d'un langage d'interrogation qui leur permet d'accéder à la B.D. et d'en dégager un certain nombre de renseignements.
- de procédures d'analyse syntaxique de la cohérence des spécifications conceptuelles,
- d'un jeu de programmes, générateurs de rapports standards de documentation et d'analyse qui peuvent les aider dans leur tâche de conception du S.I. C'est ainsi qu'ont été réalisés des outils d'édition :
  - . de structures de données sous forme de hiérarchies (Structure Report (1)),
  - . de structures de traitements ainsi que les relations dynamiques qui leur sont associées . sous forme de hiérarchies (Structure Report, Structured FPS Report (1))
    - . sous forme de tableaux (Dynamic Interaction Report (1)),
  - . des spécifications de moyens de réalisation (processeurs, ressources) (Structured FPS Report (1)),
  - . de messages d'observation lorsque des situations qui peuvent amener des problèmes dans la suite, sont découvertes: ex :
    - . lorsqu'un processus n'est déclenché par aucun événement,

---

(1) pour plus de précisions, cfr. dossiers d'analyse relatifs à ces programmes.

- lorsqu'un processus ne déclenche aucun autre processus dans la suite,
- lorsqu'un événement n'initialise aucun traitement ou séquence de processus,
- lorsque la réalisation d'un point de synchronisation n'a aucune conséquence,
- lorsque la génération d'un message n'a aucune répercussion sur le système considéré,
- lorsque l'on a découvert un cycle

(Dynamic Interaction Report (1)).

(3) Un autre type d'utilisations de la B.D. est l'évaluation de la faisabilité des spécifications conceptuelles (niveau conceptuel), compte tenu de la disponibilité d'un ensemble de ressources et de moyens (niveau logique).
Les spécifications conceptuelles, rappelons-le, décrivent les messages, les stuctures de données, les processus et le flux des informations dans ses aspects statiques et dynamiques (cfr. chap.II.1.2.).
Quant aux moyens de réalisation, nous en avons dressé l'architecture au chap.II.
Plusieurs jeux d'hypothèses sur les ressources seront testés par rapport à un ensemble de spécifications conceptuelles.

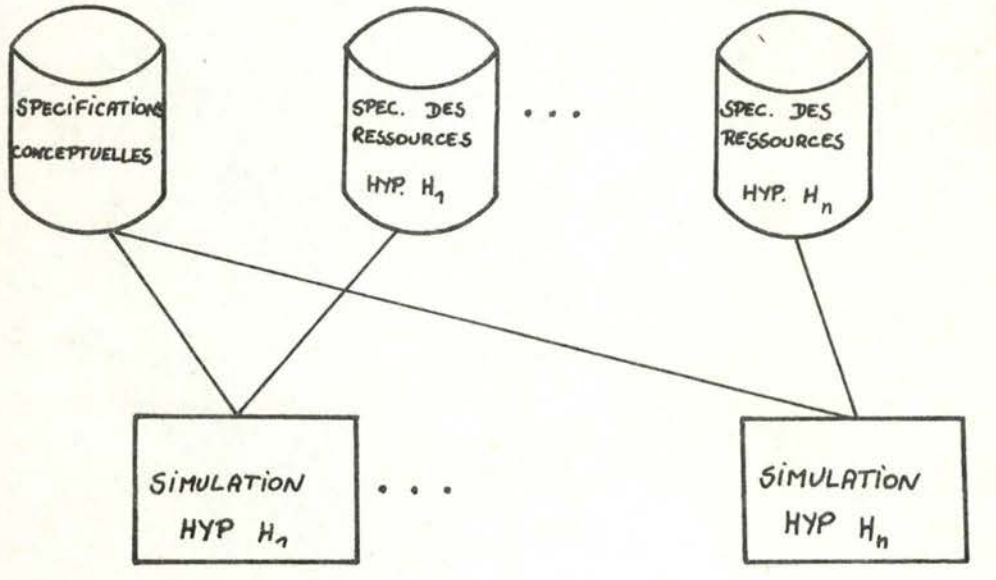


fig 43.

(1) pour plus de précisions, cfr. dossier d'analyse relatif à ce programme.

La 'faisabilité' d'un ensemble de spécifications conceptuelles sera évaluée par une analyse du comportement des ressources et du comportement des processus. Cette évaluation se fera par simulation.

C'est pourquoi on a défini :

- pour chaque type de ressources :
  - une unité d'observation,
  - une période d'analyse,
  - un ensemble de mesures (disponibilité-capacité, nombre de 'points d'entrées' à une ressource, calendrier, relations de consommation des processeurs,...);
- pour les processus :
  - une période d'analyse,
  - une unité d'observation,
  - l'hypothèse d'allocation complète des ressources (1)
  - une stratégie d'ordonnancement (niveau conceptuel des traitements avec éventuellement assignation de priorités...

Ainsi, grâce à la simulation, on pourra effectuer un certain nombre de mesures :

- pour les ressources :
  - capacité-disponibilité utilisée par période,
  - nombre de requêtes reçues;
  - distribution du temps de blocage d'une ressource,
  - ...
- pour les processus :
  - nombre d'occurrences créées,
  - temps d'attente,
  - temps de présence,
  - nombre d'interruptions,
  - temps d'interruption,

---

(1) Hypothèse d'allocation complète des ressources :
un processus déclenché ne deviendra ACTIF que si, et seulement si, les ressources qu'il requiert sont complètement allouées au début de son exécution.
De plus, il les gardera pour toute la durée de cette exécution.

. nombre de terminaisons anormales,
. ressources utilisées ou consommées,
. ...

Ces mesures peuvenr être utiles aux concepteurs du S.I.

Structure du système de simulation associé à D.S.L. (D.S.L./D.S.A.-SIMUL) (1).

L'évaluation des spécifications conceptuelles d'un système recouvre les grandes fonctions suivantes (la description des spécifications à l'aide des outils standards de D.S.L./D.S.A. étant faite) (cfr. fig.42) :

- la GENERATION qui a pour objectif de générer automatiquement, au départ de la B.D. des spécifications D.S.L., un programme exécutable de simulation (écrit en SIMULA 67) modélisant le comportement dynamique global du système,
- l'EXECUTION ou la simulation du système pendant une période déterminée et la collecte des mesures statistiques résultantes,
- l'EXPLOITATION ou analyse statique des résultats obtenus lors de la phase précédente,

chacune de ces fonctions nécessitant, lors de sa mise en oeuvre, l'intervention de l'utilisateur. Ce dernier doit en effet :

- lors de la génération, décrire l'environnement de réalisation et ses interactions avec le système étudié,
- lors de l'exécution, fournir les paramètres de contrôle de la simulation,
- lors de l'exploitation, choisir les requêtes statistiques.

Signalons, pour terminer, que les résultats obtenus lors de la simulation peuvent amener le concepteur à modifier la B.D. des spécifications si ces résultats ne le satisfont pas.

---

(1) D'après Y. PIGNEUR, auteur du système de simulation en D.S.L./D.S.A.-SIMUL .

DEUXIEME PARTIE

DEVELOPPEMENT PARTICULIER .

Chap.VI. Comment D.S.L. pourrait exprimer les concepts des modèles de MERISE.

Ce chapitre est un développement particulier des chapitres précédents. Rappelons que D.S.L. est un langage qui a pour objectif de décrire les spécifications fonctionnelles d'un S.I. à savoir :
- les spécifications conceptuelles du S.I. (cfr. chap. II 1.2.)
- la prise en compte d'un ensemble de ressources et de moyens (niveau logique, cfr. chap.II. 2.2.)

Nous allons donc nous attacher maintenant à montrer comment MERISE pourrait se servir de D.S.L. comme langage de spécification. Il suffirait simplement qu'il existe un interface entre la B.D. CATI et une B.D. de spécifications D.S.L., ce qui permettrait aux utilisateurs de MERISE de disposer des outils développés à Namur (générateurs de rapports, simulation,...).

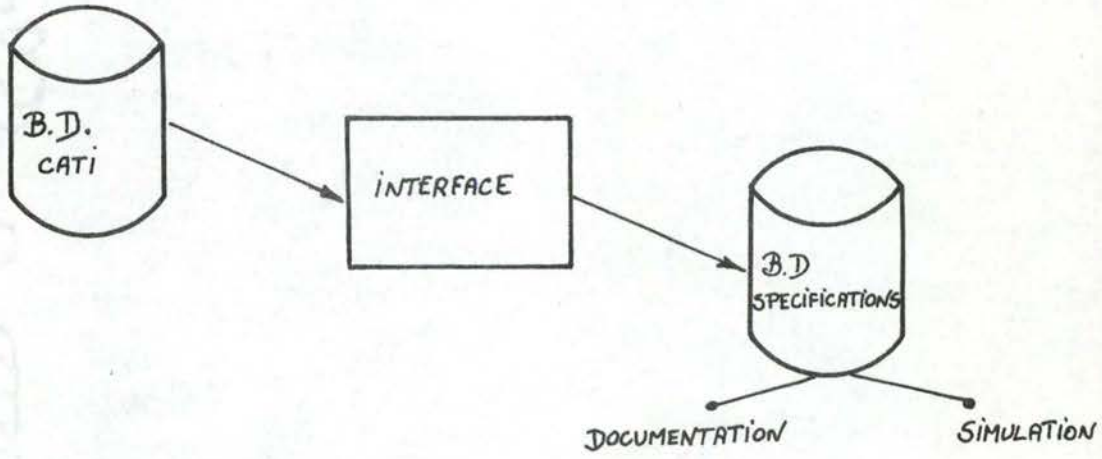


fig 44.

Il est évident qu'on pourrait imaginer un interface qui prendrait les spécifications d'une B.D. D.S.L. et les traduirait suivant les modèles de MERISE sous une forme acceptable pour le CATI. On pourrait alors bénéficier de tous les outils développés sur le CATI (cfr. chap.V. 1)

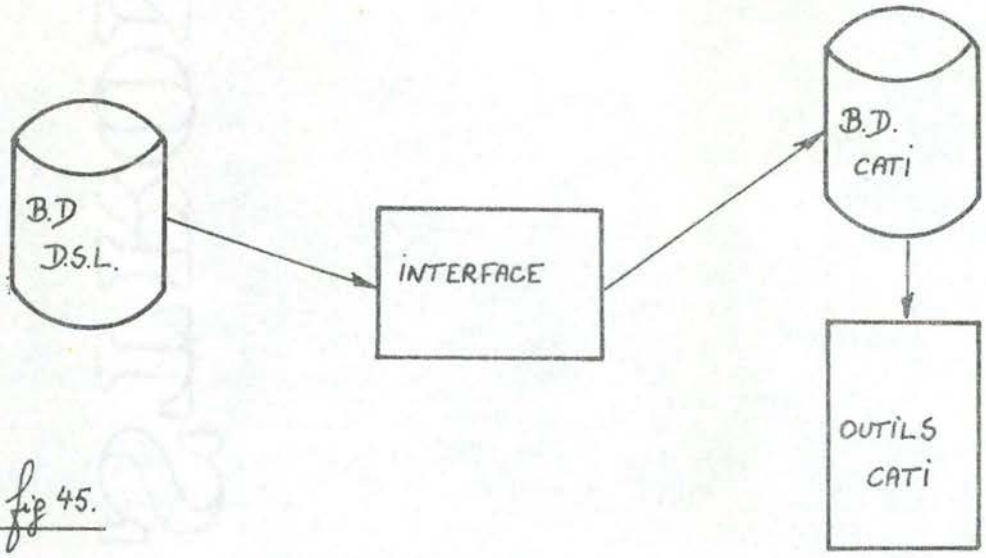


fig 45.

Nous nous contenterons ici d'étudier le 1er aspect de l'interface (fig.44.) et nous reprendrons, pour cela, les concepts des modèles - conceptuels des données et des traitements,

- logique des traitements,

modèles de MERISE qui recouvrent, plus ou moins, l'aspect fonctionnel de la description du S.I. (cfr. chap.I,24.) et nous essayerons d'établir le parallélisme avec D.S.L.

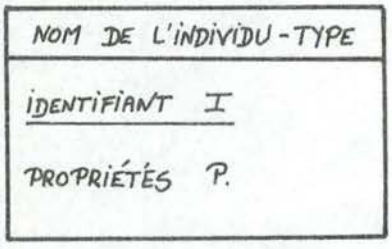
VI.1 Comment D.S.L. pourrait exprimer les concepts du M.C.D. MERISE?

.Remarque préliminaire :

Je ne rappellerai pas ici les définitions des éléments des modèles de MERISE : elles ont déjà été données au chap.II

En MERISE

Individu-type : objet concret ou abstrait du monde réel.



En D.S.L.

Entity (1) : objet décrivant un type d'information utilisé pour la représentation d'un 'type d'objet' ou d'un 'concept' du monde réel.

DEFINE ENTITY nom-de-l'individu;
IDENTIFIED BY {element-name / group-name} (,{element-name / group-name});
CONSISTS OF {element-name / group-name} (,{element-name / group-name});

D.S.L. est donc assez explicite pour exprimer les notions d'individu-type, identifiant et propriété de MERISE.
Quant aux conventions de naming et autres, elles ne posent aucun problème en D.S.L.

En MERISE
RELATION :



Rmq. : la 'collection' d'une relation-type est la liste des individus-types qui participent à cette relation.

En D.S.L.

RELATION : objet représentant une association de 2 ou plusieurs ENTITIES.

DEFINE RELATION r-nom-de-la-relation;
IDENTIFIED BY {element-name / group-name} (,{element-name / group-name});
ASSOCIATED DATA IS {element-name / group-name} (,{element-name / group-name});

Quant à la notion de 'collection', elle peut être facilement exprimée en D.S.L. grâce à la clause :
BETWEEN entity-name AND entity-name (, AND entity-name); d'une relation

---

(1) pour les définitions des concepts D.S.L. : "Dynamic Specification Language and a Model Analyzer for an Information System - Users' Manual''. F. BODART et Y. PIGNEUR.

Rmq. : on peut également, au niveau de l'ENTITY, signaler qu'elle participe à une relation par la clause :
RELATED TO entity-name (, AND entity-name)
[VIA relation-name] ;

En MERISE :

Propriété :

propriété élémentaire : plus petit élément logique d'information manipulé dans un système.

propriété composée : regroupement logique de propriétés élémentaires formant une nouvelle information sémantiquement différente de ses constituants.

En D.S.L:

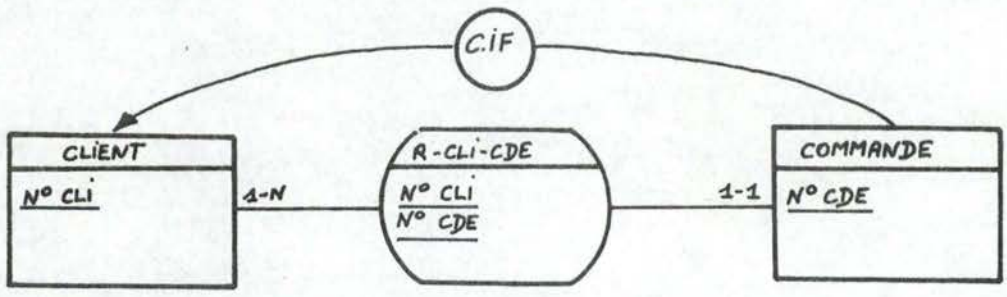
ELEMENT : objet décrivant une unité atomique d'information qui ne peut être subdivisée.

GROUP : objet décrivant une collection d'ELEMENTS et/ou GROUPS reliés logiquement entre eux.

Ces ELEMENTS et GROUPS sont rattachés aux différentes ENTITIES et RELATIONS grâce aux spécifications CONSISTS OF (CONTAINED-IN) et ASSOCIATED DATA (ASSOCIATED WITH).

En MERISE:

Cardinalité : toute relation de dimension n à n cardinalités qui définissent la participation de chacun des individus de la collection à la relation.
La cardinalité est individuelle et précise le nombre min. et max. de fois qu'une occurrence d'individu-type peut apparaître dans des occurrences de la relation-type.



En D.S.L:

Connectivity : attribut d'une relation qui permet de spécifier, pour 2 sous-ensembles de la relation : A_1 et A_2, le nombre d'occurrences de A_2 pour une occurrence de A_1.

DEFINE RELATION relation-name;
CONNECTIVITY IS syst-pm-i TO syst-pm-j
    BETWEEN entity-name AND entity-name (, AND entity-name);
Les paramètres de cette clause étant définis, par exemple, par :
DEFINE SYSTEM-PARAMETER syst-pm-i;
VALUE IS '0';

En MERISE :

Contrainte d'intégrité fonctionnelle :

indique l'existence d'une dépendance fonctionnelle entre un sous-ensemble de la collection de la relation et un individu de cette relation (individu-cible).
(à tout m-uple de la sous-collection S de dim. m ($m \leq n-1$) ne peut correspondre, dans le cadre de la relation R, qu'une seule occurrence de l'individu-cible I.)

ex : dans l'exemple ci-dessus. (cfr. cardinalité):
à une occurrence de 'commande' ne peut correspondre qu'une et une seule occurrence de 'client'.
FONCTION : commande sur client.

En D.S.L.:

Cette notion de C.I.F. n'existe pas.
Si on veut garder le parallélisme entre MERISE et D.S.L., il faudrait introduire dans D.S.L., au niveau de la relation, une clause spécifiant ces C.I.F.
Elle pourrait se formuler comme suit (pour reprendre la terminologie de MERISE) :

FONCTION IS entity-name (, AND entity-name) ON entity-name;

Pourquoi introduire cette clause?

L'intérêt d'exprimer ces C.I.F. réside dans le fait qu'elles peu-

vent notamment aider à la normalisation du M.C.D. en facilitant les décompositions de relations.
De plus, de nombreux outils opérationnels utilisent cette notion et il est donc intéressant, si on veut pouvoir reprendre ces outils d'introduire les C.I.F.

Ainsi, en ajoutant seulement cette clause de C.I.F., on est capable d'exprimer le M.C.D. (ainsi que les M.E.D. puisqu'ils utilisent les mêmes concepts) avec D.S.L.
Pour regrouper tous les concepts spécifiés en D.S.L. en une sorte de M.C.D., il suffit de spécifier un SET (objet représentant une collection d'informations conservées à l'intérieur du système) qui regroupera les entités et les relations que l'on aura définies.

Rmq : si le M.C.D. peut être spécifié par un SET, comment définir un M.E.D.?
Est-ce un SUBSET du SET 'M.C.D.'? Est-ce un SET indépendant?
Au sens mathématique, le M.C.D. ne pourrait pas être un SUBSET du SET M.C.D., car ce dernier n'est pas le résultat d'une juxtaposition de M.E.D. mais plutôt une fusion de M.E.D. (les M.E.D. n'apparaîtront plus en tant que tels dans le M.C.D.).
Cependant, si on prend SUBSET dans le sens que lui donne D.S.L., le M.E.D. peut être un SUBSET du SET M.C.D. car toutes les informations contenues dans les M.E.D. devront se retrouver dans le M.C.D. final.

VI.2 Comment D.S.L. pourrait exprimer les concepts du M.C.T. MERISE?

Rappel : pour les définitions des éléments du M.C.T., cfr.chap.II

En MERISE:

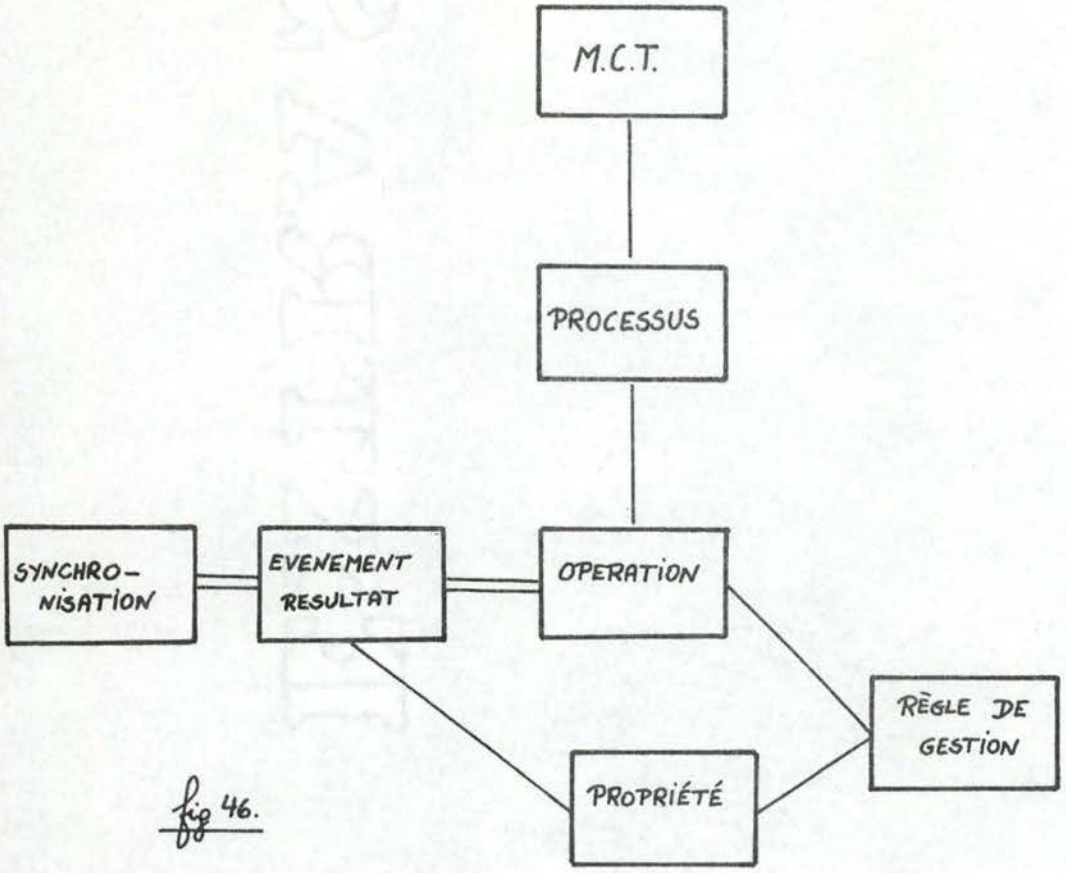


fig 46.

En MERISE :

La notion principale est celle d'opération.
De plus, différentes opérations peuvent être regroupées en un processus si elles concourent à un même but.

En D.S.L.?

Le concept principal est celui de PROCESS.
PROCESS : objet représentant des traitements d'information.
Ses conditions de déclenchement sont semblables à celles des opérations dans MERISE à savoir :

- déclenchement par un événement ou un message,
- déclenchement par réalisation d'un point de synchronisation,
- déclenchement lors d'un changement d'état d'un PROCESS (cet aspect n'est pas repris dans MERISE.)

De plus, un PROCESS est subdivisible en autant de sous-PROCESS que l'on souhaite grâce à la clause SUBPARTS ARE.

On pourra donc, grâce au concept D.S.L. de PROCESS exprimer les concepts MERISE de processus et opération :

- un processus sera spécifié par un PROCESS général dont un des attributs sera : ATTRIBUTES ARE nomenclature 'processus';
- une opération sera quant à elle spécifiée par un PROCESS (éventuellement SUBPART d'un PROCESS beaucoup plus général) mais d'action beaucoup plus limitée. Un de ses attributs sera : ATTRIBUTES ARE nomenclature 'opération';

Le fait que dans MERISE, une opération s'effectue sans attente externe et qu'elle est non interruptible ne pose aucun problème; il suffit de prendre cette convention dans les spécifications D.S.L. et éventuellement, de subdiviser les différents PROCESSES en PROCESSES plus fins et non interruptibles (les clauses INTERRUPTS / RESUMES / ABORTS } et leurs dérivés ne seront donc pas utilisées).

Signalons qu'en ce qui concerne la spécification de la durée d'une opération, la clause [DURING syst-pm] remplit cette fonction en D.S.L.

Enfin l'hypothèse d'allocation complète des ressources avant l'exécution d'un PROCESS est également faite en D.S.L.

En MERISE :

L'action exercée sur les données par une opération du système s'exprime par l'énoncé d'un certain nombre de règles de gestion. De plus, pour chacune de ces règles, on cite les propriétés en entrée et en sortie.

En D.S.L. :

Cet énoncé de REG peut très bien se faire de façon littéraire dans la PROCEDURE d'un PROCESS.

PROCEDURE : attribut d'un PROCESS qui décrit les opérations d'un PROCESS comme un ensemble de règles et d'activités reliées logiquement entre elles et spécifiées en format de texte.

ex. de PROCEDURE :

Pour un PROCESS qui calcule, à partir de données des cartes provenant d'une pointeuse, le salaire des ouvriers .

PROCEDURE;

1. calcul du salaire brut à partir de données des cartes de pointeuse,
2. calcul des taxes sur le salaire brut,
3. soustraction des taxes du salaire brut pour obtenir le salaire net,
4. mise à jour de l'enregistrement de l'ouvrier,
5. mise à jour de l'enregistrement du département,
6. génération du chèque;

Pour la spécification des informations en entrée et en sortie du PROCESS, on utilise les clauses :

RECEIVES/GENERATES pour les messages,

USES/DERIVES pour les structures de données utilisées.

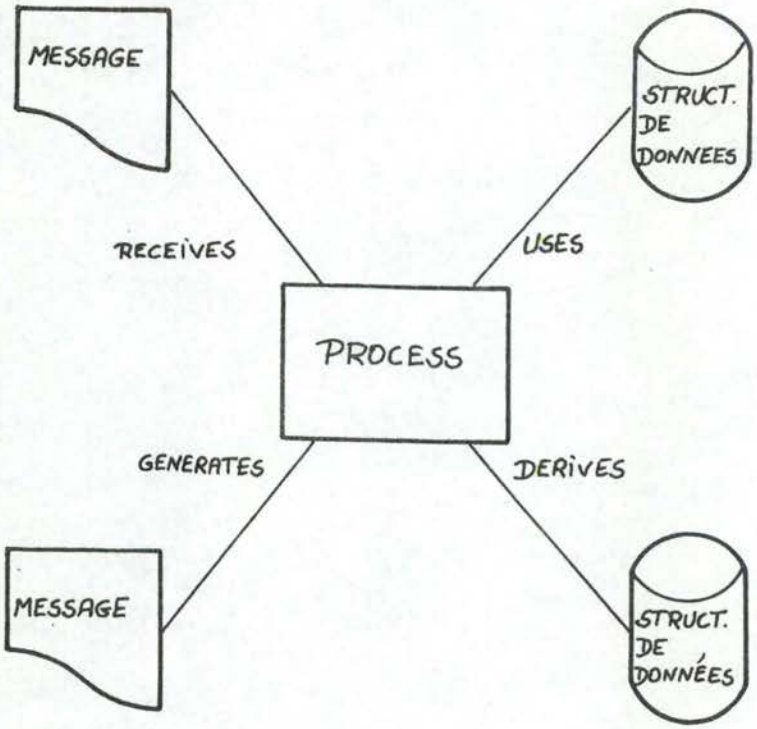
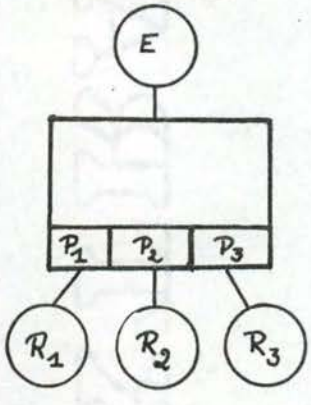


fig 47.

En MERISE :



L'émission d'un résultat par une opération est en général conditionnée par une règle d'émission, règle de gestion particulière faisant intervenir des propriétés des événements en entrée de l'opération et/ou des données mémorisées.

En D.S.L.:

Un PROCESS peut, lorsqu'il se termine (et aussi lorsqu'il change d'état), causer un événement, émettre un message, les deux pouvant éventuellement contribuer à un point de synchronisation ou même déclencher un autre PROCESS.
Toutes ces actions peuvent être effectuées en fonction de conditions que l'on peut spécifier comme suit :

IF cond-name IS [TRUE / FALSE] ON TERMINATION (TRIGGERS P; / CAUSES ev; / CONTRIBUTES TO sy; / GENERATES m;

et définir la condition comme suit :

DEFINE CONDITION cond-name;
PROBABILITY TRUE IS syst-pm;
TRUE WHILE ;

nested syntax : c'est à ce niveau que l'on donnera 'la règle d'émission'.
Cette règle peut, elle aussi, être fonction d'un attribut d'un événement en entrée ou d'un paramètre (SYSTEM-PARAMETER) défini par ailleurs

Rmq : dans la syntaxe actuelle de D.S.L., la génération d'un message et l'émission d'un événement (résultat) à partir d'un PROCESS ne sont pas soumises à la réalisation d'une condition. Il faut donc ajouter la clause [IF cond-name IS [TRUE / FALSE]] dans ces 2 cas (si c'est possible).

Donc, jusqu'à présent, la spécification d'un PROCESS peut comporter les clauses suivantes :

DEFINE PROCESS process-name;
ATTRIBUTES ARE attr-name syst-pm (, attr-name syst-pm);
RECEIVES message-name (, message-name);
USES (relation-name / entity-name / group-name / element-name) (, (relation-name / entity-name / group-name / element-name));
DERIVES (set-name / entity-name / group-name / element-name / relation-name) (, (set-name / entity-name / group-name / element-name / relation-name));
[IF condition-name IS {TRUE / FALSE}] GENERATES message-name (, message-name);

PART OF process-name;
SUBPARTS ARE process-name (, process-name);
[IF condition-name IS {TRUE / FALSE} ON {INCEPTION / TERMINATION} CAUSES event-name;
[IF condition-name IS {TRUE / FALSE} ON {INCEPTION / TERMINATION} CONTRIBUTES TO
synchro-pt-name (, synchro-pt-name);
[IF condition-name IS {TRUE / FALSE} ON {INCEPTION / TERMINATION} TRIGGERS process- name;

En MERISE :

Les événements/résultats déclenchent éventuellement par l'intermédiaire de synchronisations, les différentes opérations du M.C.T. Celles-ci émettent, lorsqu'elles sont terminées, des résultats.

En D.S.L.:

EVENT/MESSAGE :

EVENT : un événement est provoqué par un changement d'état du système connu d'un observateur uniquement par l'intermédiaire d'un message.
Le contenu du MESSAGE associé à un événement sera spécifié en tant qu'attributs de cet événement (il y a notamment, comme attributs, des précisions sur le temps et le lieu).
L'événement peut avoir 2 sortes d'attributs :
- attributs explicites spécifiés par la clause ATTRIBUTES ARE,
- attributs implicites découlant de "l'histoire" de l'événement.

On distingue également des événements externes et internes :
- événement externe : stimulus généré par un INTERFACE (1) auquel le système considéré doit réagir .
Ce stimulus sera considéré comme une occurrence de MESSAGE provenant d'un INTERFACE

(1) INTERFACE : objet décrivant une partie de l'organisation ou de l'environnement avec laquelle le système considéré est en relation par l'intermédiaire d'émission de messages.

s'il est porteur d'information :

DEFINE MESSAGE m-name;
GENERATED BY interface-name;
ON GENERATION CAUSES ev-name;

sinon, s'il est seulement un signal, il sera directement spécifié comme un EVENT explicite généré par un INTERFACE :

DEFINE EVENT ev-name;
GENERATED BY interface-name;

.événement interne : événement qui se déroule à l'intérieur du système considéré, conséquence soit d'un changement d'état d'un PROCESS, soit de l'émission d'un MESSAGE par un PROCESS.

ex : DEFINE PROCESS p;
ON TERMINATION CAUSES event-1;

ou : DEFINE MESSAGE m;
GENERATED BY p;
ON GENERATION CAUSES event-1;

Un EVENT peut évidemment déclencher un PROCESS ou contribuer à un SYNCHRONIZATION-POINT(1) (actions éventuellement soumises à la réalisation d'une certaine condition).

ex : DEFINE EVENT e;
[IF condition-name IS {TRUE / FALSE}] ON GENERATION
TRIGGERS process-name;

ou : CONTRIBUTES TO synchro-pt-name;

Rmq : en D.S.L., la notion de 'résultat' n'est pas explicite : il s'agit d'un EVENT comme les autres.
Si on veut reprendre la distinction événement/résultat de MERISE (les raisons de cette distinction ont été exposées au chap.II 1.3.), il suffira, en D.S.L., de qualifier l'EVENT qui représentera un résultat par un attribut :
ATTRIBUTES ARE nature-de-l'événement 'résultat';

En D.S.L.:

SYNCHRONIZATION-POINT : point d'attente d'une occurrence d'une

---

(1) cfr. ci-après.

combinaison d'événements.

C'est un concept tout à fait identique à celui de 'synchronisation' de MERISE.

La liste des 'entrées' au point de synchronisation sera donnée par la clause de CONTRIBUTED BY.

Le prédicat sera explicite lors de la spécification du PREDICATE du SYNCHRONIZATION-POINT (2).

C'est aussi au niveau du PREDICATE que l'on fournira les conditions locales grâce à la clause WHERE.

Chaque réalisation d'un point de synchronisation génère une occurrence d'un certain événement (qui peut être implicite); ce dernier aura comme attributs implicites, les attributs des événements contribuants.

Par définition, chaque flux menant au point de synchronisation disparaît en tant que tel et est fondu dans un nouveau flux lors de la réalisation de cette synchronisation.

La contribution d'un élément d'entrée à la synchronisation peut être limitée en durée : elle peut être instantanée, elle est effectuée jusqu'à la réalisation du point de synchronisation ou elle peut être limitée par l'émission d'un certain événement (c'est l'interêt de la notion 'durée de vie' d'un événement/résultat dans MERISE) :

clause : CONTRIBUTES TO synchro-pt-name
UNTIL ...., de l'EVENT.

Ainsi, on peut avoir :

```
DEFINE SYNCHRONIZATION-POINT synchro-pt-name;
CONTRIBUTED BY event-name (UNTIL...);
CONTRIBUTED BY GENERATION OF message-name (UNTIL...);
CONTRIBUTED BY {TERMINATION  OF process-name;
               {INCEPTION
CONTRIBUTED BY synchro-pt-name UNTIL...;
CAUSES event-name;
CONTRIBUTES TO synchro-pt-name UNTIL... ;
TRIGGERS process-name;
```

---

(2) au niveau de l'expression booléenne dans le PREDICATE, il serait intéressant de pouvoir utiliser '('et')'.

PREDICATE;
nested syntax;

Cette syntaxe est simplifiée : d'autres clauses peuvent, en effet, être spécifiées (conditions, délais,...).

En MERISE :

Commentaires sur les règles d'évolution.

Consommation des occurrences d'un événement (résultat au niveau de la synchronisation (1)).

Une fois le prédicat vérifié, la synchronisation est activée et instantanément franchie.

Les occurrences en entrée nécessaires à la vérification de p sont alors consommées et l'événement interne en sortie généré.

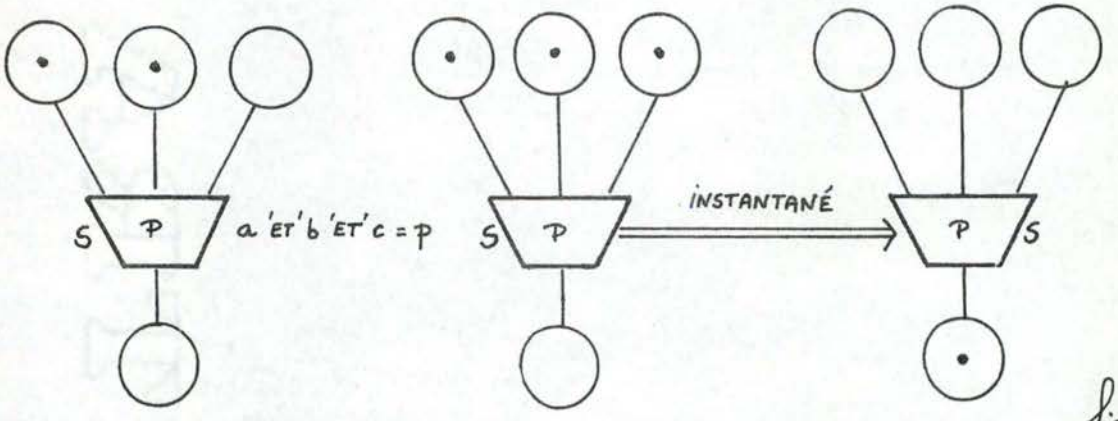


fig 48

Rmq : il est possible de conditionner le franchissement de la synchronisation S à une variable supplémentaire : variable d'état 'majeure' provenant d'un moniteur; ceci permet d'activer un ensemble de synchronisations avec des priorités différentes (cfr. chap.V.1).

En D.S.L.:

Dès que le prédicat est vérifié, le point de synchronisation est "réalisé" et également franchi instantanément.

Quant au conditionnement du franchissement d'un point de syn-

---

(1) Les règles de consommation des occurrences des événements/résultats lors du franchissement de synchronisations ou d'opérations s'inspirent du mécanisme de consommation des jetons dans les réseaux de Pétri.

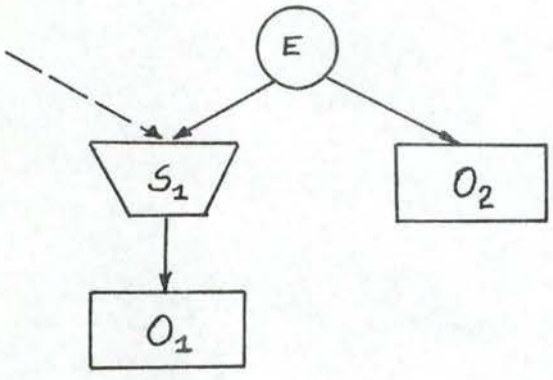
chronisation en introduisant une variable 'majeure' gérée par un moniteur, cela n'apparaît pas explicitement en D.S.L. Des politiques d'ordonnancement sont cependant introduites au niveau de la simulation, notamment pour tester des politiques de réalisation. Un moniteur s'avère alors nécessaire pour gérer les problèmes de priorités et les règles d'ordonnancement.

En MERISE :

Commentaires sur les conflits, les cardinalités d'un événement/résultat, et la participation d'un événement/résultat.

Lorsqu'un type d'événement externe (ou de résultat interne) participe à plusieurs synchronisation(s) et/ou opération(s), on dit qu'il y a conflit possible pour ou sur ce type d'événement (ou de résultat).

ex.



CONFLIT SUR L'EVENEMENT E

fig 49

Le conflit est résolu si les conditions d'entrée de ces événements dans les synchronisations sont exclusives.

ex.

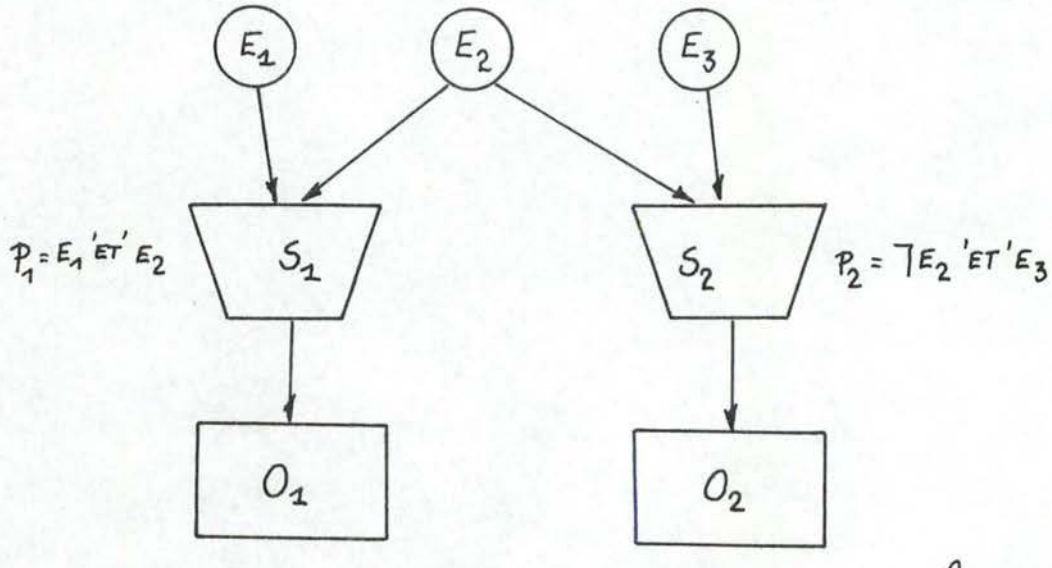


fig 50.

On peut également éliminer le conflit sur un événement ou un résultat en les produisant avec une cardinalité convenable c.-à-d. en les produisant sous la forme de plusieurs occurrences identiques (plusieurs jetons), autant en fait que de synchronisation(s) et/ou opération(s) en conflit.

ex : reprenons la fig.49

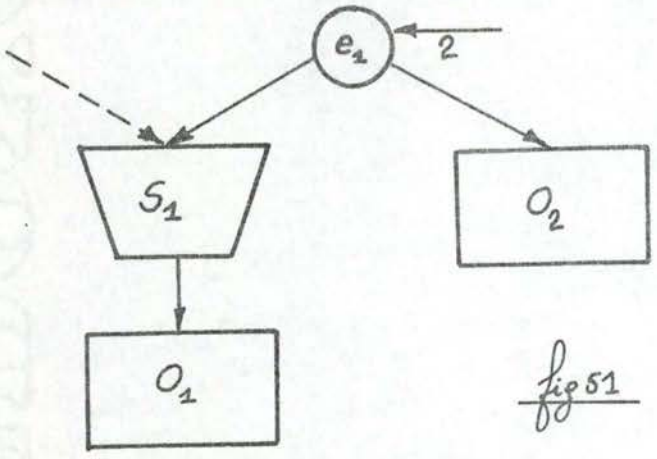


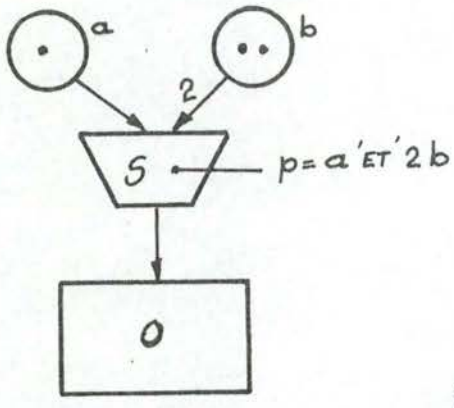
fig 51

plus de conflit sur e_1 car 2 occurrences identiques sont produites.

Rmq : Toute occurrence produite devra alors être consommée. Si, par le jeu des cardinalités, on ne parvient pas à régler le conflit, le moniteur le fera en utilisant ses variables d'état.

On appelle participation ou largeur de l'événement-type E ou du résultat-type R à la synchronisation S (ou l'opération O), le nombre d'occurrences différentes de E ou de R nécessaires à l'activation de S(ou de O).

ex :



participation de b à S : 2

fig 52.

En D.S.L.:

MERISE, pour le formalisme des traitements, se base sur les réseaux de Pétri. Chaque occurrence d'événement et/ou résultat

sera assimilée à un jeton qui 'voyagera' dans le graphe en suivant certaines règles de consommation et de génération (ex : lorsqu'une synchronisation ou une opération est activée, il y a consommation d'une occurrence (éventuellement n) du ou des type(s) d'événements/résultats nécessaire(s) pour cette activation).

Les notions de conflit et de cardinalité découlent directement de cette symbolisation. En D.S.L., on n'utilise pas cette schématisation, ce qui simplifie le problème. En effet, une occurrence d'événement peut, par exemple, déclencher plusieurs PROCESSES et participer à plusieurs SYNCHRONIZATION-POINTS sans que cela pose des problèmes particuliers de conflit et de cardinalité : il suffit simplement de signaler l'activité de cet événement lors de sa spécification.

Quant à la notion de participation, elle a été résolue en D.S.L. en utilisant un point de synchronisation qui utilise dans la clause PREDICATE un compteur qui, dès que le nombre fixé d'occurrences d'événement est atteint, fait que le point de synchronisation est réalisé.

Ex : déclenchement d'une occurrence de PROCESS de facturation pour 100 formules de commande reconstituées.

```
DEFINE SYNCHRONIZATION-POINT sy-accumulation;
PREDICATE;
        COUNT (ev-reconstituted-order) = 100;
```

COUNT est une fonction qui compte le nombre de réalisations des points de synchronisation sy-order-reconstitution et donc le nombre d'événements ev-reconstituted-order.

VI.3. Comment D.S.L. pourrait exprimer les concepts du M.L.T. MERISE.

Rappel : les définitions des concepts du M.L.T. de MERISE ont été données au chap.II.2.1.

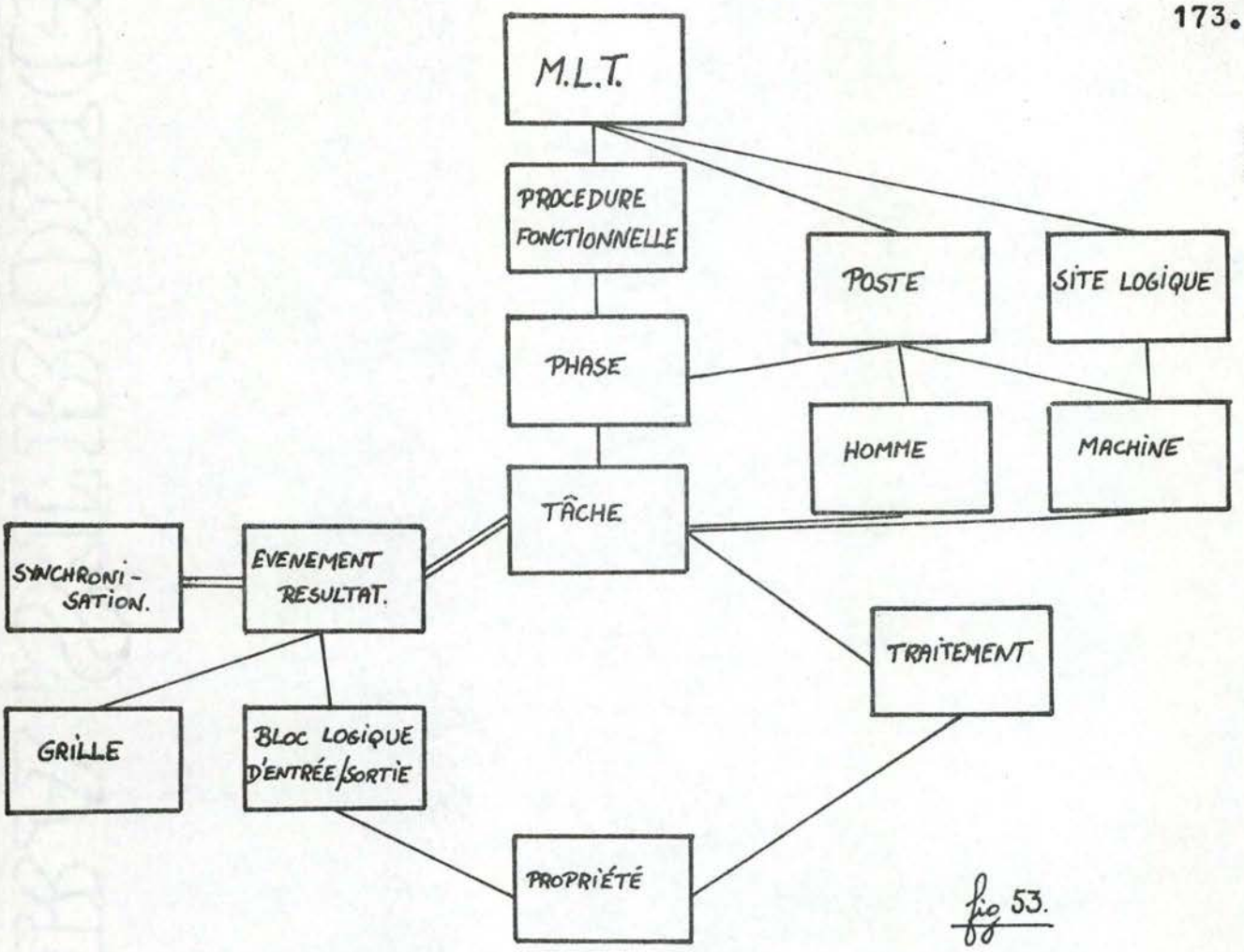


fig 53.

En MERISE :

Comme nous avons déjà eu l'occasion de le signaler, la structure logique des traitements (M.L.T.) de MERISE est fort hiérarchisée :

procédure fonctionnelle-phase-tâche-traitement.

En D.S.L.:

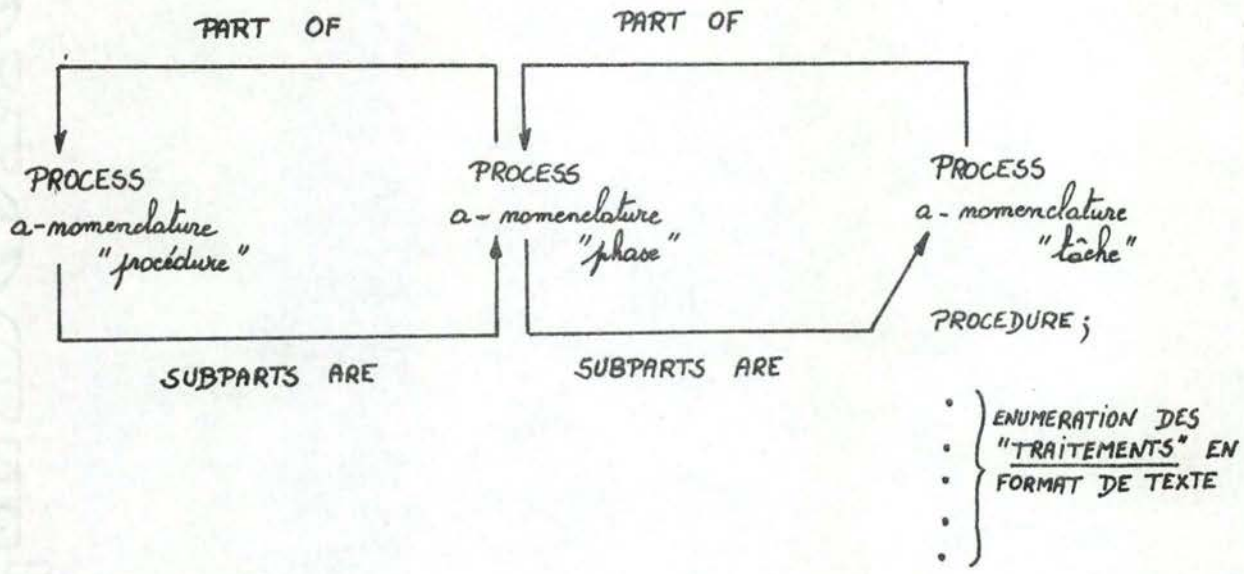
Remarque préliminaire :

Afin de toujours savoir à quel niveau on se situe dans les modèles de MERISE, il serait utile de préciser, pour chaque objet D.S.L. spécifié, un attribut 'de niveau' :

ATTRIBUTES ARE a-level 'conceptual';

ou ATTRIBUTES ARE a-level 'logical';

En ce qui concerne la structure procédure-phase-tâche-traitement, on peut la représenter en D.S.L. comme suit :



La procédure fonctionnelle MERISE sera traduite en D.S.L. de la façon suivante : (1)

```
   DEFINE PROCESS pf-nom-de-procédure;
   ATTRIBUTES ARE a-level 'logical',
                  a-nomenclature 'procedure';
   SUBPARTS ARE t-nom-de-phase,
                t-nom-de-phase,
                  ....        ;
   TRIGGERED BY GENERATION of nom-de-MESSAGE;
   RECEIVES m-nom-de-MESSAGE;
ou TRIGGERED BY nom-d'-EVENT;
   GENERATES nom-de-MESSAGE;
   ON TERMINATION CAUSES nom-d'-EVENT;
```

En ce qui concerne les phases de MERISE, on les traduit en D.S.L. par :

```
   DEFINE PROCESS ph-nom-de-phase;
   ATTRIBUTES ARE a-level 'logical',
                  a-nomenclature 'phase';
   PART OF pf-nom-de-procédure;
   SUBPARTS ARE t-nom-de-tâche,
                t-nom-de-tâche,
                  ...         ;
   TRIGGERED BY nom-d'-EVENT;
ou TRIGGERED BY nom-de-synchro-point;
```

(1) Il s'agit d'une syntaxe simplifiée .

ou TRIGGERED BY GENERATION OF m-nom-de-MESSAGE;
RECEIVES m-nom-de-MESSAGE;
ON TERMINATION CAUSES ev-nom-d'-EVENT;
et/ou
GENERATES m-nom-de-MESSAGE;

Les tâches et les traitements deviendront en D.S.L. :

DEFINE PROCESS t-nom-de-tâche;
ATTIBUTES ARE a-level 'logical',
a-nomenclature 'tâche';
PART OF ph-nom-de-phase;
TRIGGERED BY ev-nom-d'-EVENT;
ou TRIGGERED BY nom-de-synchro-pt;
ou TRIGGERED BY GENERATION OF m-nom-de-MESSAGE;
RECEIVES m-nom-de-MESSAGE;
ON TERMINATION CAUSES ev-nom-d'-EVENT;
et/ou
GENERATES m-nom-de-MESSAGE;
PROCEDURE;

- énumération en format de texte de l'ensemble des opérations effectuées par la tâche, c.-à-dire l'ensemble des 'traitements' qu'elle réalise.
- Cette description peut déjà être orientée vers la programmation en particulier si elle définit un algorithme plus ou moins précis (utilisation de structures telles que : si...alors...sinon... ;
  recommencer l'opération jusqu'à ce que...);

En MERISE :

On retrouve au niveau du M.L.T., les concepts d'événement, de résultat et de synchronisation que l'on a déjà définis au M.C.T.

Rmq. : Les synchronisations interviennent éventuellement en début de phase ou de tâche mais pas en début d'une procédure fonctionnelle puisque cette dernière est déclenchée par une occurrence d'un type d'événement (naturel).

De plus, pour les événements ou les résultats, on peut décrire un certain nombre de blocs logiques d'entrée ou de sortie et grilles.

En D.S.L. :

La traduction en D.S.L. des événements, des résultats et des synchronisations de MERISE, ne posent aucun problème grâce aux concepts d'EVENT et de SYNCHRONIZATION-POINT déjà vus plus haut (niveau conceptuel des traitements).

En ce qui concerne :

- les blocs d'entrée ou de sortie qui ne sont en fait que des ensembles de propriétés rattachées soit à des événements soit à des résultats, on peut facilement en tenir compte en D.S.L. grâce à la clause CONSISTS OF rattachant un ensemble de propriétés (GROUP) à un MESSAGE en entrée ou en sortie d'un PROCESS (un GROUP par BLE ou BLS)

  On aura ainsi, par exemple :

  DEFINE MESSAGE message-name;
  ...
  CONSISTS OF group-name (, group-name);

  un GROUP étant défini de la façon suivante :

  DEFINE GROUP group-name;
  CONSISTS OF {group-name / element-name} (, {group-name / element-name})
  CONTAINED IN message-name;

- les grilles : leur description peut se faire facilement sous forme de commentaires grâce à la clause LAYOUT d'un objet MESSAGE;

  DEFINE MESSAGE message-name;
  LAYOUT;
  comment entry;

En MERISE :

Aux concepts déjà vus s'ajoutent ceux de

poste : centre d'activité élémentaire, de l'organisme, il comprend tout ce qui est nécessaire (hommes, machines, espace, outillage...) à l'exécution des tâches.

site : ensemble de moyens de traitement automatisé établis en un lieu donné.

En D.S.L.:

Le poste sera défini comme étant un objet PROCESSOR, objet qui peut exécuter des PROCESSES (en l'occurrence des PROCESSES ph-nom-de-phases) en utilisant et en consommant des ressources. Les PROCESSORS 'postes' peuvent être composés, tout comme en MERISE, d'hommes et de machines que l'on spécifiera : (en tant que SUBPARTS OF...)

- soit comme ressources réutilisables (REUSABLE-RESOURCES); dans ce cas, on aura tendance à privilégier la notion de 'poste';
- soit comme processeurs auxiliaires utilisés par le poste afin d'accomplir certaines 'tâches' (au sens MERISE) bien définies.

C'est cette seconde proposition qui me semble la mieux adaptée ici car elle exprime une certaine activité (→PROCESSORS) de la part des hommes et des machines.

Rmq : Il n'y a pas de différence de nature entre un processeur et une ressource réutilisable mais une différence de niveau d'analyse.

La spécification du poste pourrait donc être en D.S.L. :

```
DEFINE PROCESSOR pr-nom-de-poste;
UNIT OF MEASURE IS...;
SUBPARTS ARE pr-nom-de-processeur auxiliaire (, pr-nom-de-
processeur auxiliaire);
ACTIVE FOLLOWING...;  (CALENDRIER D'ACTIVATION).
PERFORMS ph-nom-de-phase;
EMPLOYS pr-employé AT RATE OF...
          TO PERFORM t-nom-de-tâche-1;
EMPLOYS pr-machine-1 AT RATE OF...
          TO PERFORM t-nom-de-tâche-2;
...
CONSUMES.....;   (SPECIFICATION DES RESSOURCES CONSOMMÉES).
```

avec

```
DEFINE PROCESSOR pr-employé;
UNIT OF MEASURE IS...;
PART OF pr-nom-de-poste;
CAPACITY IS.X.SHARABLE AMONG.X.;   X: NBRE D'EMPLOYES INTERCHANGEABLES.
ACTIVE FOLLOWING...;
PERFORMS t-tâche-1 DURING...;
```

```
PERFORMS t-tâche-4 DURING...;
EMPLOYED BY pr-nom-de-poste AT RATE OF...
        TO PERFORM t-nom-de-tâche-1;
...
CONSUMES...;
```

de même pour le PROCESSOR auxiliaire pr-machine-1.
Quant au site, on pourrait le définir comme un processeur englobant un certain nombre de processeurs auxiliaires (ou de ressources réutilisables) de type 'machine' :

```
DEFINE PROCESSOR site;
SUBPARTS ARE pr-machine-1,
             pr-machine-2,
             ...;
```

VI.4. Prise en compte des qualifications et des ressources.

Nous venons donc de voir comment on peut traduire en D.S.L. les concepts des modèles de MERISE suivants :

- modèles conceptuels des données et des traitements,
- modèle logique des traitements,

ce qui contitue en fait une grande partie des spécifications fonctionnelles du S.I.
Il reste maintenant à prendre en compte les quantifications et les ressources.

Les quantifications :

Les quantifications sont contenues dans le CATI. Leur traduction en D.S.L. ne pose aucun problème : on dispose en effet de toutes les clauses nécessaires.

Ex : .cardinalités de relations :

```
CONNECTIVITY IS sys-pm TO sys-pm BETWEEN...
```

.nombre d'occurrences d'individus :

```
CARDINALITY IS sys-pm  FOR entity-name  (; sys-pm
  FOR entity-name  );
```

clause définie au niveau d'un SET.

.des fréquences de génération de messages ou d'événements/résultats :

```
GENERATED BY {process-name  } AT FREQUENCY OF sys-pm
             {interface-name}
TIMES PER interval-name;
```

.de calendriers ou d'intervalles de temps :

DEFINE CALENDAR calendar-name;

ou DEFINE INTERVAL interval-name;

.des paramètres de tous genres :

DEFINE SYSTEM-PARAMETER sys-pm;

VALUE IS...;

.des unités de mesure et de prix :

DEFINE UNIT unit-name;

-...

Les ressources.

Pour que les spécifications fonctionnelles soient complètes il faudrait y inclure des spécifications propres aux ressources.Cela permettrait,entre autres, à MERISE d'utiliser de façon intéressante les outils de simulation développés sur la B.D. de spécifications D.S.L.

Malheureusement, si les processeurs sont bien considérés dans MERISE (postes - hommes - machines), les notions de ressources ne le sont pas (du moins pas encore).

Le concepteur MERISE devra donc introduire lui-même les spécifications D.S.L. relatives aux ressources (REUSABLE-RESOURCES, CONSUMABLE-RESOURCES) ainsi que les clauses qui les détaillent (processeurs qui les utilisent ou les consomment, taux d'utilisation et de consommation...).

Conclusion.

Synthèse du mémoire.

Les systèmes d'information automatisés sont amenés à jouer un très grand rôle dans le 'management' des organisations. Des méthodes de conception et de réalisation de S.I. voient le jour un peu partout. Nous avons essayé, durant ce travail, de comparer deux de ces méthodes : MERISE, développée à Aix en Provence par l'équipe d' H. TARDIEU, et 'NAMUR', étudiée à l'Institut d'Informatique de Namur par l'équipe de F. BODART.

Cette comparaison s'est faite en plusieurs étapes. Après avoir rappelé les différents niveaux qui ont été retenus par MERISE et 'NAMUR' pour la description des S.I. (niveaux conceptuel, logique et physique), nous avons exposé les modèles de données et de traitements propres aux niveaux de description conceptuel et logique (le niveau physique ne nous a pas intéressé dans ce travail car il est trop tributaire de considérations techniques.

Nous avons étudié ensuite, pour chacune des deux approches retenues, les correspondances qui ont été établies entre les différents modèles (modèles externes → modèles conceptuels, modèles conceptuels → modèles logiques) ainsi que les problèmes de cohérence entre les modèles de données et les modèles de traitements et ce, à chaque niveau.

Les démarches méthodologiques pour la conception du S.I. ont fait également l'objet d'un chapitre. Il faut cependant signaler que la démarche proposée par 'NAMUR' n'en est encore qu'à l'état embryonnaire.

Nous avons considéré ensuite les outils développés par MERISE et 'NAMUR. Ces outils travaillent à partir d'une B.D. que l'on aura eu soin de construire (CATI ou B.D. de spécifications) et ont pour but d'aider le ou les concepteur(s) du S.I. dans sa (leur) tâche de conception et de réalisation.

Enfin, en guise d'annexe, nous avons jeté les bases d'un interface qui, à partir du CATI MERISE, pourrait opérer la traduction en D.S.L. et fournir une B.D. de spécifications sur laquelle travailleraient les outils développés par 'NAMUR'.

Intérêt du mémoire.

Ce mémoire avait notamment pour objectif de proposer une démarche méthodologique propre à l'Institut d'Informatique de Namur et qui serait le résultat de la mise en commun d'un ensemble de travaux déjà effectués par certaines personnes de l'Institut. Elle s'appuierait sur des modèles et proposerait un certain nombre d'outils d'aide au concepteur. Nous avons baptisé cette méthode : 'NAMUR' et nous l'avons décrite durant tout ce travail. De plus, nous l'avons comparée à MERISE afin de dégager les similitudes et les différences qui existent entre ces deux méthodes, aussi bien au niveau des concepts que nous avons tenté de clarifier, que de la démarche et des outils.

Nous en avons également profité pour montrer les travaux faits au niveau de la description de la dynamique (traitements) dans les S.I. et ce, à travers l'étude des deux approches considérées. Nous avons vu ainsi comment il était possible de modéliser les traitements du S.I. en conservant la découpe en niveaux conceptuel, logique et physique, découpe utilisée depuis longtemps dans la construction des bases de données.

Nous avons pu aussi constater la similitude des travaux effectués par les équipes d'Aix en Provence et de Namur, à une exception près : le développement des outils. En effet, si MERISE s'efforce d'élaborer des outils de manipulation du CATI en vue de la réalisation du S.I., 'NAMUR' a donné la priorité au développement d'un langage de spécification (incluant l'aspect dynamique du S.I.) et d'un outil qui doit permettre d'évaluer, par simulation, la 'faisabilité' des spécifications conceptuelles effectuées, par rapport à un ensemble de ressources et de moyens.

C'est en raison de cette similitude que nous avons envisagé de construire un interface qui permettrait à MERISE d'utiliser les outils développés à Namur.

Extensions possibles.

Pour chacune des deux méthodes, il reste à développer un certain nombre d'outils :

Pour MERISE :

Il reste à élaborer .la plupart des outils concernant les traitements du S.I. (construction et vérification du M.C.T., passage au M.L.T., passage au M.P.T., ...);
.un outil de simulation du fonctionnement du S.I. (cfr. 6. p.137).

Pour 'NAMUR' :

On envisage de construire .les outils qui doivent faciliter l'exploitation de la B.D. où sont stockés les résultats de la simulation (cfr. fig.42 p.150);
.un générateur automatique d'une 'maquette' de programme ('throw - away' code) à partir de la spécification de l'algorithme afin d'évaluer le caractère effectif des règles de traitement et de l'algorithme.

Enfin, toujours dans l'optique de la comparaison, il resterait à évaluer l'efficacité des deux démarches quant à la conception et à la réalisation des S.I. On pourrait prendre pour cela, une application particulière, construire le S.I. en utilisant MERISE et 'NAMUR' et en comparer les résultats.
Pourquoi pas même reprendre le cas 'PETITPAS' et le traiter suivant MERISE? Ce travail était cependant trop long pour être fait dans le cadre de ce mémoire.

Je terminerai en soulevant quelques questions :
-Quelle est, en effet, la productivité de telles méthodes, notamment pour les P.M.E.? N'impliquent-elles pas l'utilisation d'un matériel puissant et d'un personnel fort qualifié dont elles ne disposent pas?
-Quelle est l'efficacité des concepts relatifs à la dynamique des S.I. Reflètent-ils bien la réalité?
Ces questions feront peut-être l'objet d'autres travaux!

Bibliographie .

Le MOIGNE J.L. : "La théorie du système général. Théorie de la modelisation", Systèmes - Décisions.

NIJSSEN G.M. : "Modelling in Data Base Management Systems", IFIP, North Holland Publishing Company, 1975.

ROLLAND C., LEIFERT S., RICHARD C. : "Le pilotage de l'évolution des S.I.", Inforsid, Aix en Provence, Janvier 1980.

Références MERISE.

TARDIEU H., HECKENROTH H. : "Conception des systèmes d'informations automatisées - MERISE : Guide pratique pour l'élaboration des modèles, outils privilégiés de la démarche." CETE Aix en Provence, Décembre 1979.

TARDIEU H., HECKENROTH H. : "Conception des S.I.A. - Glossaire pour les données et les traitements." CETE Aix en Provence, Janvier 1979.

TARDIEU H., HECKENROTH H. : "Conception des systèmes d'informations automatisées - MERISE : L'étude préalable - Un cas pratique de conception de solution.", CETE Aix en Provence, Decembre 1979.

TARDIEU H., HECKENROTH H. : "Conception des systèmes d'informations automatisées - MERISE : Présentation générale.", CETE Aix en Provence, Octobre 1979.

TARDIEU H., HECKENROTH H. : "Conception des systèmes d'informations automatisés - MERISE : Transparents 2ème partie.", CETE Aix en Provence, Octobre 1979.

TARDIEU H., HECKENROTH H., ESPINASSE B. : "Niveaux de modèles, formalismes et outils pour la dynamique dans les S.I.", Séminaire INFORSID gr. II, Aix en Provence, Janvier 1980.

TARDIEU H., HECKENROTH H., NANCI D. : "Méthode, modèles et outils pour la conception de la base de données d'un S.I. - Le manuel de référence." GRASCE - CETE Aix en Provence, Décembre 1976.

TARDIEU H., NANCI D. : "Méthode, modéles et outils pour la conception d'un S.I. - Le passage au modèle interne", GRASCE - CETE Aix en Provence, Novembre 1978.

TARDIEU H., NANCI D., PASCOT D. : "Conception d'un S.I. - Construction de la base de données.", Edition de l'organisation, 1979.

Références 'NAMUR'.

BODART F. : "Analyse du cas PETITPAS - casus, spécifications D.S.L. et rapports de documentation.", Institut d'Informatique, Namur, 1979.

BODART F., PIGNEUR Y. : "Dynamic Specification Language and a Simulation Model Analyzer for an Information System - Users' Manual", Institut d'Informatique, Namur, 1979.

BODART F., PIGNEUR Y. : "Evaluation de la "faisabilité" d'un ensemble de spécifications fonctionnelles", Institut d'Informatique, Namur, Janvier 1980.

BODART F., PIGNEUR Y. : "Modèles et outils d'aide à l'analyse des spécifications fonctionnelles d'un système d'information", Institut d'Informatique, Namur, 1980.

BODART F. : "Problèmes d'organisation et méthodes d'analyse fonctionnelle", cours Institut d'Informatique, Namur, 1979.

Clarinval A. : "Méthodologie de l'analyse", cours Institut d'Informatique, Namur, 1977.

HAINAUT J.L. : "Un modèle de description de fichiers : le modèle d'accès.", Institut d'Informatique, Namur, 1980.

Hainaut J.L. : "Analyse du cas PETITPAS - Dossier d'analyse organique - Première partie : la base de données", cours Institut d'Informatique, Namur, 1979.

BUMP
0035906999
*FM B16/1980/09