Le Projet MACSI - 1

Alfred de Chelminski

THESE

présentée à

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

POUR OBTENIR LE GRADE DE

DOCTEUR ~~D'ETAT~~ Ingénieur

Alfred de CHELMINSKI

LE PROJET MACSI - 1

Thèse soutenue le 30 juin 1975 devant la commission d'examen :

Président : Monsieur L. BOLLIET

Examinateurs : Monsieur J.C. BOUSSARD
Monsieur C. DELOBEL
Monsieur J.C. DERNIAME

Rapporteur : Monsieur F. PECCOUD

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

M. Michel SOUTIF	Présidents	M. Louis NEEL
M. Gabriel CAU	Vice-Présidents	MM. Lucien BONNETAIN, Jean BENOIT

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

PROFESSEURS TITULAIRES

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

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

PROFESSEURS ASSOCIES

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

PROFESSEURS SANS CHAIRE

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

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

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

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

MAITRES DE CONFERENCES ASSOCIES

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

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

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

Fait à Saint Martin d'Hères, OCTOBRE 1974.

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

PROFESSEURS TITULAIRES

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

PROFESSEUR ASSOCIE

M. BOUDOURIS Georges	Radioélectricité

PROFESSEURS SANS CHAIRE

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

MAITRES DE CONFERENCES

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

MAITRE DE CONFERENCES ASSOCIE

M. LANDAU Ioan Doré	Automatique

CHARGE DE FONCTIONS DE MAITRES DE CONFERENCES

M. ANCEAU François	Mathématiques appliquées

Fait à Saint Martin d'Hères JANVIER 1974

Mes remerciements vont en premier lieu à

Monsieur François PECCOUD, Chargé des fonctions de Maître de Conférences à l'Institut Universitaire de Technologie de Grenoble, qui a eu le courage, ou l'inconscience, de m'accepter au sein de son équipe de recherches pour m'initier, dans un premier temps, aux problèmes de l'informatique appliquée à la gestion, et par la suite, pour m'intégrer dans l'équipe du projet MACSI.

Pour ses idées qui sont à l'origine de cette thèse, pour l'aide qu'il m'a apportée dans sa réalisation et la patience avec laquelle il a suivi mon travail et surtout pour l'amitié dont il m'a témoignée, je tiens à lui exprimer ma plus profonde reconnaissance.

Je tiens aussi à remercier ceux qui, par leurs remarques, ont contribué à l'aboutissement de ce travail :

Monsieur le Professeur Louis BOLLIET, qui m'a fait l'honneur de présider le jury ;

Monsieur Jean-Claude DERNIAME, Maître de Conférences à l'Université de Nancy et Monsieur Claude DELOBEL, Maître de Conférences à l'Université de Grenoble, qui ont bien voulu accepter de faire partie du jury ;

Monsieur le Professeur Jean-Claude BOUSSARD, qui s'est vivement intéressé à ce travail et qui a formulé des critiques précises et très constructives.

Je voudrais exprimer également ma reconnaissance :

à Jean-Pierre GIRAUDIN, pour sa participation active, désintéressée et permanente dans ce travail, ainsi qu'aux autres membres de l'équipe MACSI : Dominique JAHU et Michel LARCHER, dont les conseils ont grandement amélioré le fond et la forme de cette thèse ;

au groupe d'Ingénieurs de l'I.F.C. (*) : Pierre BLANGERO, Jean-Michel PERNET, Jean-François ROCH et Jean-Marie SOLDANO, réalisateurs du projet MEDOC, version opérationnelle qui a permis de valider les principaux concepts de MACSI-1 ;

au département Informatique de l'I.U.T. de Grenoble et particulièrement à son Directeur, Monsieur Jean SCHEID, qui m'ont donné l'occasion d'acquérir une expérience pédagogique au sein de l'équipe d'analyse.
Je n'oublie pas Xavier CASTELLANI, qui m'a guidé et encouragé dans mes débuts en informatique de gestion ;

à l'Instituto Universitario de Tecnologia de CARACAS et son Directeur Monsieur Federico RIVERO PALACIO, qui ont rendu possible mon séjour en France et qui m'ont offert constamment leur confiance.

Enfin, je voudrais remercier

Marie-José DOREL, pour le soin, la grande compétence et la sympathie avec lesquels elle a dactylographié ce travail, ainsi que le service de reprographie, qui a assuré la réalisation matérielle de cette thèse.

(*) Institut de Formation et de Conseil - I.U.T.-B
Place Doyen Gosse - GRENOBLE

SOMMAIRE

SOMMAIRE

AVANT-PROPOS

Nous ne voulons pas, dans le cadre de ce travail, présenter au lecteur un manuel de référence ni un manuel d'utilisation, encore moins un dossier technique de programmation permettant d'assurer la maintenance des programmes de MACSI-1.

Notre propos est de présenter de façon aussi complète que possible les principes, méthodes et moyens essentiels utilisés pour réaliser MACSI-1.

Nous ferons cette présentation par étapes successives :

. *Le chapitre 1 décrira le "modèle MACSI-1"* ; il regroupera les principaux concepts qui sont à la base du modèle d'analyse proposé. Une bonne connaissance de ces concepts permet ensuite la présentation d'un "langage d'analyse", c'est-à-dire d'un langage de spécification d'un projet qui permet de manipuler selon des règles très strictes ces différents concepts. Nous terminerons ce chapitre par l'utilisation de ce langage pour assurer la description fonctionnelle d'une application.

. *Le chapitre 2 sera une description fonctionnelle de MACSI-1 par MACSI-1* : un cas d'utilisation de ce langage au niveau de sa propre description fonctionnelle. Nous aurions pu prendre n'importe quelle application connue par tout informaticien (application de paye ou de gestion des stocks par exemple) mais ce choix aurait été plus ou moins arbitraire et il n'aurait pas apporté les éléments suffisants au niveau de la progression dans la connaissance du produit MACSI-1. Aussi, avons-nous préféré utiliser le langage de description fonctionnelle de MACSI-1 pour décrire fonctionnellement MACSI-1 lui-même et permettre ainsi de préciser sa mise en oeuvre. Ainsi, dans ce cas, le produit d'analyse MACSI-1 est à la fois outil d'analyse et objet de l'analyse.

. *Dans le chapitre 3, nous définirons le "Langage de description organique"* : description des éléments du langage disponibles pour spécifier les choix techniques d'implémentation d'un projet : description des programmes, des fichiers, d'une base de données.

. *Chapitre 4 : Une méthode d'implémentation des programmes disponibles sous MACSI-1* : Concernant les programmes de gestion qui permettent d'exploiter les spécifications rédigées dans le langage d'analyse de MACSI-1, présentés aux chapitres précédents, nous évoquerons les principaux choix techniques retenus pour les réaliser. Il ne s'agit pas d'une description exhaustive de tous les programmes disponibles. Nous décrivons :

- comment avec le logiciel de base de données Socrate, nous avons organisé la gestion des spécifications stockées en mémoire,
- quelles sont les normes de programmation les plus importantes retenues pour l'implémentation de ces programmes,
- les possibilités d'évolution de la méthode disponible dans MACSI-1 et les moyens offerts pour la formation du personnel qui envisage une utilisation de cette méthode.

. *Chapitre 5 : Les Conclusions*

Enfin, nous évoquerons dans ce chapitre les réflexions que nous nous faisons aujourd'hui quant aux possibilités d'intégrer l'outil MACSI-1 dans un enseignement de l'analyse en informatique de gestion.

oOo

CHAPITRE 1

DESCRIPTION GENERALE DE MACSI-1

1.a. Introduction

Il existe sur le marché de l'informatique appliquée à la gestion de nombreuses méthodes d'analyse commercialisées par différentes sociétés de service ou constructeurs d'ordinateurs. Nous pouvons citer, à titre d'exemple : TAG, ADS, MINOS, CORIG, LCP, LCS, CANTOR, etc .... La liste que nous présentons ici n'est pas exhaustive et les méthodes qu'elles utilisent n'ont pas toutes les qualités requises pour une bonne description d'un système d'information ou une construction d'une base de données.

Le lecteur peu familiarisé avec ces méthodes trouvera dans les références suivantes ([9], [18], [21], [25])
des études comparatives de ces méthodes, sur les possibilités qu'elles apportent et leurs limites. Il est hors de notre propos ici de reprendre ces études.

Cependant, le grand nombre de ces méthodes et la disparité de leurs possibilités ne permettent pas d'affirmer que sont résolus tous les problèmes fondamentaux de conception d'un système informatique pour la gestion. On trouvera dans les références suivantes ([3], [7])
une liste des principaux problèmes qui n'ont pas encore trouvé aujourd'hui de solutions satisfaisantes.

MACSI-1 est une méthode de conception des systèmes d'informations appliqués à la gestion, composée d'un certain nombre de programmes d'aide à la conception.

Avant de préciser quels ont été les objectifs prioritaires ayant orienté la réalisation de cette méthode, il semble souhaitable d'exposer d'abord d'une façon très brève le fonctionnement général de sa mise en oeuvre et de donner un nom à ses principaux constituants.

L'utilisation de MACSI-1 peut être résumée dans le schéma de la figure 1.1.

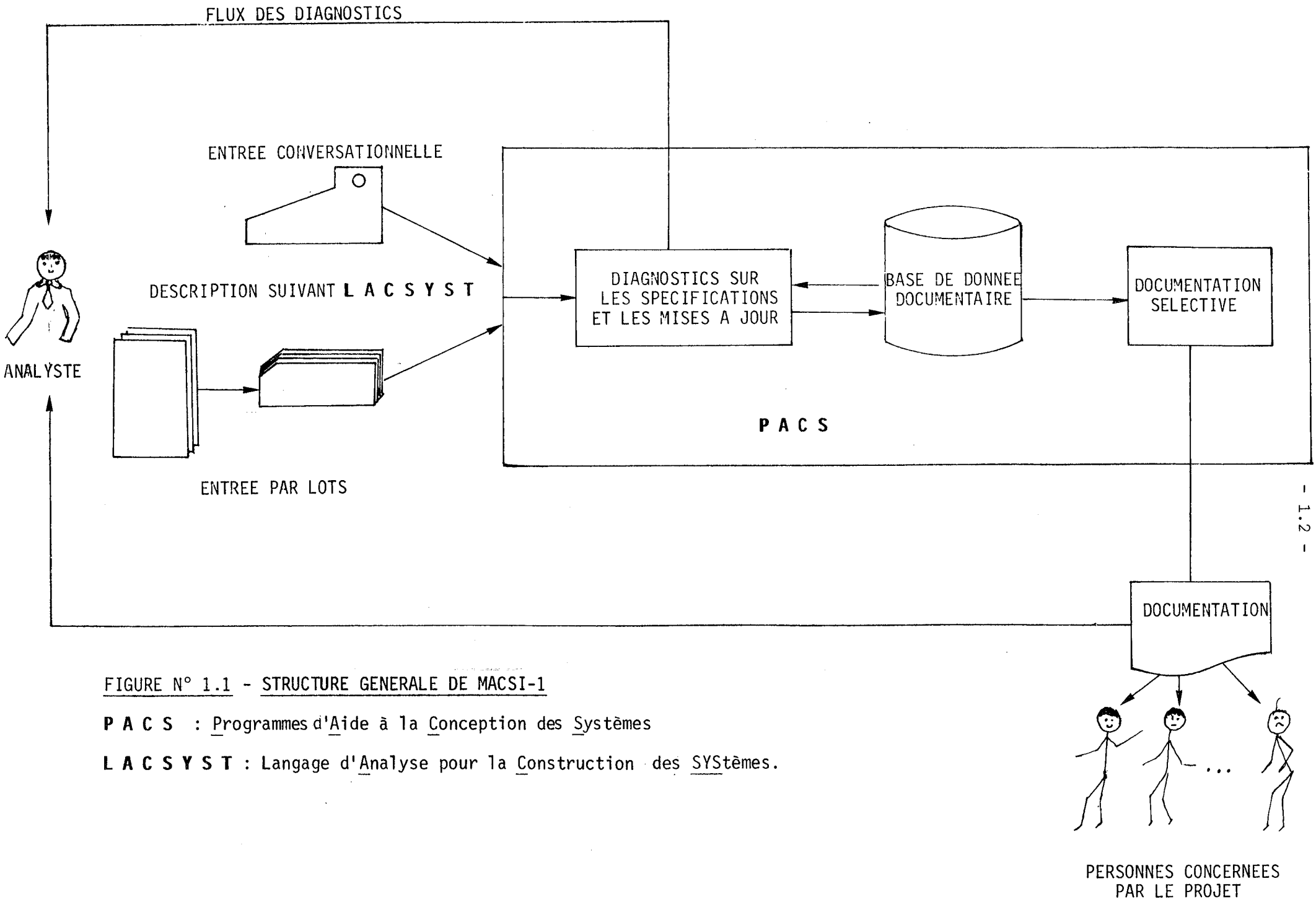


FIGURE N° 1.1 - STRUCTURE GENERALE DE MACSI-1

P A C S : Programmes d'Aide à la Conception des Systèmes

L A C S Y S T : Langage d'Analyse pour la Construction des SYStèmes.

Les analystes qui participent à un projet disposent du langage LACSYST (Langage d'Analyse pour la Construction de SYSTèmes) qui permet de décrire un projet depuis la phase "d'analyse fonctionnelle" jusqu'à la phase d'implémentation de ce projet, en passant par la phase dite "d'analyse organique". Les spécifications écrites dans ce langage sont introduites en machine, soit en mode conversationnel à partir d'une console, soit en mode "batch" sous forme de cartes perforées. Elles sont prises en compte par des programmes de diagnostic qui analysent si elles sont complètes et cohérentes. En cas d'anomalies, des diagnostics sont édités pour permettre à l'analyste de corriger ses spécifications initiales. Ce cycle se repétant jusqu'à ce que ces programmes de diagnostic acceptent ces spécifications, et dans ce cas, mettent à jour une BASE DE DONNEE DOCUMENTAIRE (BADODO). Cette base de donnée documentaire contient toutes les spécifications concernant le projet qui ont été acceptées par les programmes de diagnostic. Elle est accessible à tout moment par des programmes de DOCUMENTATION SELECTIVE qui permettent de diffuser, à la demande, tout ou partie du contenu de la base de donnée documentaire aux différentes PERSONNES CONCERNEES PAR LE PROJET. L'ensemble des programmes d'aide à la conception opérant sur la base BADODO, que ce soient les programmes de diagnostics ou les programmes de documentation automatique, représentent le logiciel d'aide à la conception de systèmes PACS (Programmes d'Aide à la Conception des Systèmes).

Ce schéma général se retrouve dans d'autres méthodes d'aide à l'analyse et, en particulier, dans le cadre du projet ISDOS. On pourra comparer les fonctions de LACSYST dans MACSI-1 et le langage PSL du projet ISDOS (réf. [32]) ou bien comparer les programmes de PACS avec les programmes PSA du projet ISDOS.

Connaissant le fonctionnement général de MACSI-1, nous pouvons maintenant définir quels en sont les objectifs prioritaires.

1.b. Les objectifs prioritaires de MACSI-1

1.b.1. Distinguer clairement l'analyse fonctionnelle et l'analyse organique d'un projet

Il est en général admis de distinguer deux phases principales dans la réalisation d'un projet informatique : la phase "d'analyse fonctionnelle" et la phase "d'analyse organique et de réalisation". Rappelons brièvement les différentes tâches qui doivent être réalisées pendant ces deux phases.

* Dans la phase "d'analyse fonctionnelle", le chef du projet informatique, assisté de spécialistes représentant les principaux services de l'administration ou de l'entreprise concernée par le projet, doit d'abord procéder à une analyse de la situation existante, c'est-à-dire des procédures manuelles ou automatisées de traitement de l'information qui seront plus ou moins modifiées, supprimées ou transformées par le nouveau projet informatique. Cette analyse de l'existant permet de faire un inventaire :
- des documents déjà utilisés dans l'organisation,
- des différents circuits de ces documents,
- des procédures de contrôle, de mise à jour et d'exploitation des fichiers alimentées par ces documents de saisie de l'information.

Une fois l'analyse de l'existant terminée, le chef de projet doit définir une solution, c'est-à-dire un ensemble de procédures qui permettraient d'atteindre les objectifs définis par la direction ayant demandé la réalisation du projet. Cette solution doit être définie FONCTIONNELLEMENT en ce sens que les procédures doivent être spécifiées sous forme de fonctions de traitement de l'information, les circuits doivent être indiqués sous forme de relations fonctionnelles entre services indépendamment des choix organiques, techniques, qui permettront de réaliser ces fonctions.

* La phase "d'analyse organique et de réalisation" a pour tâche de choisir les meilleurs moyens et méthodes techniques permettant une implémentation efficace des fonctions retenues dans la solution proposée à l'issue de la phase d'analyse fonctionnelle. Lorsque les choix techniques sont tous

clairement établis, une équipe de spécialistes (des informaticiens pour la programmation, des organisateurs pour la mise en place de nouveaux circuits administratifs) réalisent puis testent ces propositions techniques.

Cependant, l'expérience prouve que cette distinction entre analyse fonctionnelle et analyse organique, si elle est souhaitable au niveau de la réussite du projet, est cependant rarement obtenue dans la pratique quotidienne. La plupart des projets contiennent en effet, au niveau de la solution élaborée à la fin de la phase d'analyse fonctionnelle, des propositions qui ne sont pas exemptes, tant s'en faut, de choix organiques, quelquefois implicites, mais qui masquent par leur présence les choix spécifiquement fonctionnels qui ont été réalisés. En effet, le choix des fonctions à réaliser indépendamment des moyens matériels à mettre en oeuvre doit être fait dans un souci essentiel d'adéquation de ces fonctions aux objectifs qui ont été retenus. Si la nécessité oblige les analystes à prendre en compte certains problèmes techniques dans la définition de la solution fonctionnelle, on constate qu'ils sont irrémédiablement tentés de rendre prioritaires ces choix techniques par rapport au souci de faire coïncider la solution avec les objectifs retenus. Pour s'en convaincre, il suffit de constater que dans le langage PSL (Problem Statement Language) du projet ISDOS, en plus des spécifications fonctionnelles, d'autres sont en fait organiques comme, par exemple, la définition de la longueur des zones d'information ou la structure de certains documents (réf. [32]).

Le tableau 1.2. montre la séparation que nous pouvons envisager entre les aspects fonctionnels et organiques à propos des décisions qui doivent être posées dans un projet informatique.

	Aspects fonctionnels	Aspects organiques
Choix des données	Propriétés, types de valeur, contrôles, relation avec d'autres données	Longueur, représentation interne en mémoire.
Choix des fichiers		Description de l'enregistrement concerné, support, blocage, volume, organisation. clé.
Choix des programmes	Décomposition de l'application en unités fonctionnelles de traitements	Découpage des unités fonctionnelles en unités de programmation. Pour chaque programme, on définit le type (saisie, contrôle, tri, édition), le langage de programmation utilisé, l'espace des données, les fichiers concernés.
Choix des documents	Description externe des documents, type d'entrée ou de sortie	Dessin du document, espace des données, contrôles.

TABLEAU 1-2

A partir de ce constat, nous nous proposons d'offrir aux analystes, dans MACSI-1, un langage de description, LACSYST, qui permette de distinguer clairement les spécifications fonctionnelles des spécifications organiques.

1.b.2. Obtenir des futurs utilisateurs du projet informatique une participation active dans la spécification fonctionnelle de ce projet

La situation la plus courante à l'heure actuelle dans l'organisation de l'équipe chargée de réaliser un projet est celle où l'ensemble des membres qui participent à la rédaction des spécifications sont tous des informaticiens. Dès lors, le chef de projet, les analystes-programmeurs utilisent un formalisme très spécialisé pour spécifier leur projet, formalisme incompréhensible pour quelqu'un n'ayant pas une formation informatique approfondie. Ces informaticiens réalisent leurs spécifications après avoir interrogé plus ou moins les différents futurs utilisateurs du projet dont ils ont la charge. Leur opinion, en général, est que ces utilisateurs savent fort mal quels sont leurs bescins, et n'ont aucune idée, même approximative, des possibilités et des

limites d'un ordinateur. Il existe en général un véritable problème de dialogue entre ces deux catégories de personnes. Pour essayer de résoudre ce problème, sont quelquefois organisés des cours d'initiation à l'informatique pour ces utilisateurs ; ces cours sont insuffisants pour permettre de participer de façon active à la rédaction des spécifications du projet.

Dans le cadre de MACSI-1, toute une partie de la phase d'analyse fonctionnelle est confiée entièrement aux futurs utilisateurs du projet. Plus précisément, ils disposent dans LACSYST d'un sous-ensemble du langage qui leur permet de définir complètement :

- l'organisation administrative dans laquelle sera inséré le projet informatique,
- les données permanentes qui doivent être manipulées dans les différents documents et fichiers,
- ainsi que, sous une forme approximative, les principaux traitements qu'ils jugent nécessaires pour contrôler les données en entrée ou pour produire les données en sortie du système informatique.

1.b.3. Proposer des mécanismes de construction d'un projet informatique cohérent avec ceux adoptés dans le cadre des efforts pédagogiques de programmation structurée

De différentes expériences pédagogiques d'enseignement de l'analyse que nous avons eues, il ressort que de graves difficultés subsistent au niveau de l'efficacité de cet enseignement et de son assimilation par les étudiants. Cet enseignement associe en général des présentation de cas, des expériences pratiques brèves sous forme de travaux pratiques ou longues sous forme de réalisation d'un projet, et un cours qui regroupe un certain nombre de recommandations sur les normes à respecter et les erreurs à ne pas commettre. Il est souvent perçu dans ces conditions par les étudiants comme un agrégat de recettes dont l'assimilation dépend essentiellement de la durée et du bon sens de chacun, conditionné par une formation élémentaire. L'enseignement de l'analyse semble être à l'heure actuelle dans l'état où était la programmation lorsque programmation était synonyme d'apprentissage d'un langage. Si hier l'enseignement de la programmation se restreignait souvent à la rédaction de programmes en Fortran, Cobol ou Algol, aujourd'hui l'enseignement de l'analyse se confond le plus souvent avec l'apprentissage d'une méthode d'analyse associée à de nombreux exercices.

Dans cette perspective, il faut constater que les efforts d'enseignement entrepris depuis plusieurs anées pour construire des cours d'algorithmique indépendants de tout langage de programmation ont porté leurs fruits et que les élèves informaticiens commencent à acquérir les mécanismes fondamentaux de la programmation. Nous voulons évoquer en particulier tous les efforts qui ont été réalisés dans le domaine de la programmation structurée (réf. [8], [10], [20]). Il nous est apparu prioritaire de proposer dans le cadre de MACSI-1 des mécanismes analogues de construction d'un projet informatique qui dégagent les principes fondamentaux de construction d'un système. Nous avons en particulier retenu l'obligation faite aux analystes de spécifier leur projet suivant un processus d'analyse descendante identique à celui proposé dans le cadre des enseignements de programmation structurée. Cette obligation de réflexion en partant du général pour aboutir finalement aux éléments les plus détaillés du projet permet en particulier de combattre le défaut, trop souvent constaté chez les étudiants, de se noyer dès le démarrage d'une étude informatique dans une multitude de détails contradictoires qui n'ont place dans aucun schéma d'ensemble.

1.b.4. Utiliser au maximum l'ordinateur pour tester la cohérence des spécifications produites et faciliter leur diffusion

Aux programmes réalisés pour gérer de façon automatique la documentation produite dans le cadre d'un projet informatique, nous avons fixé dans MACSI-1 trois objectifs prioritaires :

1.b.4.1. Contrôler la cohérence des spécifications

Cette documentation, en effet, quel que soit le projet, est très volumineuse. Rédigée sur papier, elle occupe plusieurs centaines de pages dont bon nombre sont des descriptifs de fichiers, de données, de programmes, de circuits de l'information très complexes. L'aspect combinatoire de cette documentation (correspondance entre données et fichiers, entre fichiers et programmes qui les utilisent) permet difficilement d'assurer la vérification manuelle de la cohérence des spécifications produites.

Il est aisé de vérifier par programme que les conditions suivantes sont réalisées dans les spécifications d'un projet :

- Contrôler que tous les codes d'identification des données, des fichiers, des programmes, des procédures administratives, etc .. sont tous discriminants, c'est-à-dire qu'à un code donné ne correspond pas plusieurs objets différents dans la description du système informatique.

- Vérifier que les spécifications sont complètes, c'est-à-dire que par rapport au modèle d'analyse retenu, certaines précisions n'ont pas été omises.

- Contrôler la cohérence par rapport à un critère d'utilité qui peut porter soit sur les données (vérifier que toutes les données stockées dans les fichiers ont au moins une utilisation au niveau des traitements de sortie), soit au niveau des documents de sortie (vérifier que ces documents de sortie ont au moins un utilisateur).

- Vérifier de façon automatique le respect du principe "ne pas mettre la charrue avant les boeufs !". On pourra vérifier en particulier par programme que l'ordre dans lequel les spécifications sont fournies ne conduit pas à donner des précisions organiques sur certaines parties de l'application alors que l'ensemble de l'analyse fonctionnelle, par exemple, n'est pas terminée.

- Il doit être enfin possible de contrôler de façon automatique le respect du planning prévisionnel de réalisation du projet et de mesurer les écarts entre ce planning et l'avancement constaté par l'introduction des spécifications.

- D'autres critères de cohérence pourront être précisés ultérieurement.

1.b.4.2. Procédures de documentation sélective

Sans originalité par rapport à d'autres méthodes de documentation assistée par ordinateur, il faut bien envisager la possibilité offerte par la machine d'assurer une diffusion sélective de la documentation d'un projet à partir du moment où celle-ci est entièrement stockée sur des fichiers magnétiques. Cette possibilité de documentation automatique sélective doit diminuer les coûts de reproduction des dossiers. Elle doit aussi faciliter la mise à jour des spécifications au fur et à mesure des modifications introduites.

1.b.4.3. Offrir des possibilités de modification faciles et rapides des programmes de gestion de la documentation dès que le modèle d'analyse proposé dans MACSI-1 est jugé partiellement inadéquat pour contrôler un projet particulier

Il est en effet évident qu'une méthode d'analyse quelle qu'elle soit ne permet pas de prendre en compte la gestion de n'importe quel projet informatique. Selon les contraintes principales du projet ou ses objectifs essentiels (problèmes de coûts, problèmes de délais de réalisation, problèmes de portabilité de la réalisation, ...), la documentation doit pouvoir être adaptée à la nature du projet et faire ressortir les éléments sur lesquels porteront les contrôles fondamentaux du projet. Ainsi, partant de la constatation d'une nécessaire adaptation d'une méthode d'analyse à la nature du projet auquel elle est appliquée, il nous a paru essentiel d'offrir la possibilité de modifier facilement les programmes associés à la méthode d'analyse MACSI-1. C'est pourquoi, comme nous le verrons au chapitre 4, les principes généraux d'implémentation des programmes de PACS, ainsi que la structure de données BADODO ont été fixés en retenant comme prioritaire le critère de facilité de modification plutôt qu'un critère de rapidité d'exécution des programmes.

1.c. Le modèle de système d'informations

Dans une organisation, plusieurs systèmes se superposent : système de production, système financier, système de gestion du personnel, système de marketing, etc. Bien qu'apparemment ils soient indépendants, il existe en réalité de nombreuses relations et interactions entre eux. C'est le système d'information qui fait la liaison entre tous les autres systèmes, de la même manière que le système nerveux assure la commande et la coordination des fonctions vitales et la réception de messages sensoriels des systèmes circulatoire, moteur, osseux, etc... Les applications en informatique de gestion ne sont qu'une partie du système d'informations de l'entreprise (ou de l'administration).

Pour représenter un système d'informations, nous avons besoin d'un modèle. Ce modèle doit nous permettre de faire les descriptions :
- des différentes fonctions élémentaires du système,
- des ressources utilisées par le système pour les assurer,
- et des interactions entre les différents constituants du système.

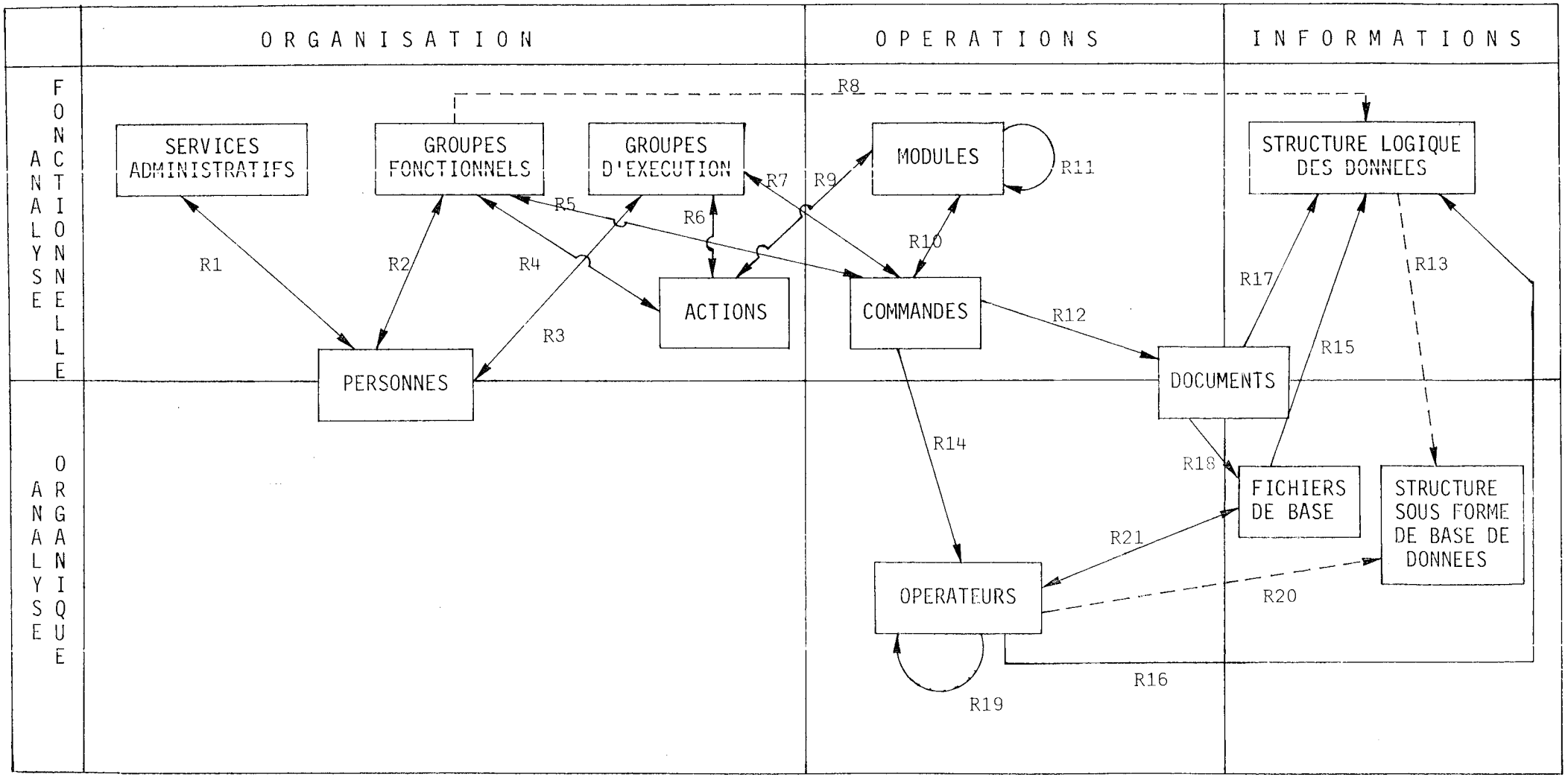
Examinons ici la structure générale du modèle retenu dans MACSI-1 pour décrire le système d'informations d'une famille assez large d'organisations.Ce modèle ne prétend pas être universel, c'est-à-dire qu'il correspondrait à toutes les situations, mais il faut pouvoir le faire évoluer, compte tenu des problèmes spécifiques de chaque organisation, des contraintes dues au matériel informatique, des habitudes issues d'une certaine histoire, etc ... (réf. [22], [24]).

1.c.1. Description qualitative du modèle

Le schéma de la figure 1-3 résume les différents types de constituants du modèle que nous distinguons dans un projet informatique et les relations existant entre ces constituants.

Dans les projets, nous distinguons (cf. figure 1-3) :

Les PERSONNES appartenant à l'organisation et qui ont un rôle dans l'application informatique. En effet, ces PERSONNES sont rattachées à l'un des SERVICES ADMINISTRATIFS de l'organisation (relation R1 de la figure) et dans le cadre de l'application informatique, elles peuvent appartenir à deux types de groupe (R2, R3) :



LEGENDE :

——————→ Relation explicite

– – – – – – → Relation implicite

FIGURE 1.3 - MODELE GENERAL DE SYSTEME D'INFORMATIONS

* *GROUPES FONCTIONNELS* : constitués en grande partie par les futurs utilisateurs de l'application, sont chargés de spécifier fonctionnellement le nouveau système. Ils utiliseront le langage de spécification LACSYST pour décrire en particulier les traitements automatiques de l'information (R5), les données associées (R8) et les actions (R4) en amont et en aval de l'ordinateur qui devront être exécutées pour un bon fonctionnement du projet.

* *GROUPES D'EXECUTION* : une fois l'application opérationnelle, ces groupes accompliront les différentes ACTIONS (R6) et lanceront l'exécution des différents programmes de traitement de l'information associés aux COMMANDES (R7).

Il est nécessaire d'introduire cette distinction entre les groupes chargés des spécifications et ceux chargés de l'exécution car, dans la réalité, pour une tâche donnée, ces deux groupes se confondent rarement.

Une personne fait partie de l'équipe de conception du projet (appartient à l'un des GROUPES FONCTIONNELS) ou bien elle est utilisatrice du produit opérationnel associé à l'application (appartient à l'un des GROUPES D'EXECUTION). Evidemment, une personne peut appartenir à la fois à un ou plusieurs groupes fonctionnels et à un ou plusieurs groupes d'exécution.

Notons que nous faisons la différence entre les services administratifs et les groupes, qu'ils soient fonctionnels ou d'exécution. En effet, un service administratif est une unité hiérarchique de l'organisation qui, en tant que telle, peut ne pas avoir de signification fonctionnelle dans le projet. Un groupe fonctionnel ou d'exécution est défini par rapport aux tâches prises en compte dans le cadre du projet informatique. Ainsi, un groupe peut être :

- soit une partie d'un service administratif formé par des personnes qui ont une même responsabilité dans le projet,
- soit un ensemble de personnes appartenant à différents services administratifs qui, pour ce projet informatique, ont les mêmes rôles de conception ou d'exécution.

* Les *ACTIONS* regroupent toutes les opérations qui font partie intégrante du projet et ne sont pas des traitements automatisés de données. Par exemple, on peut considérer comme ACTIONS les opérations de décision, de

fabrication, de contrôle technique ou financier, de traitement de l'information non automatisée, etc ... Autrement dit, dans un projet informatique les ACTIONS nous permettent de décrire toutes les opérations en amont et en aval de l'ordinateur qui ont une importance dans le projet lui-même.

* Les COMMANDES sont des demandes de traitement automatisé de l'information. Elles sont spécifiées logiquement par les différents groupes fonctionnels sous forme de texte libre sans tenir compte des choix techniques qui, au niveau organique, permettront de l'implémenter. Au niveau fonctionnel, le système informatique est perçu comme une boîte noire dans laquelle on entre par des commandes d'entrée et de contrôle et dont on sort par des commandes de sortie. Toutes les opérations intermédiaires que l'on fera ultérieurement après différents choix techniques : des tris, des sauvegardes de fichiers, des réorganisations de base de données, ..., ne sont pas à prendre en compte au niveau fonctionnel. Ces commandes sont précisées en grande partie par les futurs utilisateurs de l'application généralement non informaticiens, mais appartenant à l'un des groupes fonctionnels. Ils ont donc deux choses à dire : nous alimentons le système informatique avec des données ou nous voulons obtenir du système informatique des données après traitement.

* La communication entre l'homme et le système informatique se fait à travers des DOCUMENTS (imprimés, listings, messages sur écran cathodique, messages envoyés par un clavier, bordereau de perforation) en entrée ou en sortie du système. Il existe une relation fonctionnelle entre le document et la commande qui le traite (R12) et entre le document et la structure logique de données (R17) par les informations contenues.

* Commandes et actions ne s'exécutent pas dynamiquement dans le projet dans n'importe quel ordre : les groupes fonctionnels ou de conception décriront des MODULES qui sont des assemblages d'actions (R9) et des commandes (R10) ; les règles d'assemblage, précisées ultérieurement, spécifient dans quel ordre les actions et les commandes seront exécutées. Ces groupes fonctionnels devront aussi décrire les "données" concernées par l'application. Ils le feront fonctionnellement sous forme d'une STRUCTURE LOGIQUE DES DONNEES (R8) qui permet la description des informations et des relations sans tenir compte des structures de fichiers ou des bases de données qui seront retenues au niveau organique, c'est-à-dire au niveau de la programmation

du projet. Tous ces constituants sont spécifiés dans une première phase d'analyse fonctionnelle et seront complétés par des informaticiens au niveau organique. A ce niveau, MACSI-1 permet de définir :

* Une *STRUCTURE SOUS FORME DE BASE DE DONNEES* déduite à partir de la structure logique de données (R13).

* Des programmes écrits dans un langage permettant la gestion de cette base de données.

- Un système de gestion de base de données est l'ensemble des programmes assurant :
  - . la structuration
  - . le stockage
  - . la mise à jour
  - . la recherche

de l'information dans une base de données et l'interface avec les différentes utilisations de ces données.

* Les *FICHIERS DE BASE* déduits aussi de la structure logique de données (R15) sont des structures de données traitées par des programmes écrits en langages tels que COBOL, PL/1,ASSEMBLEUR, etc ...

- Un fichier est une collection d'enregistrements de même nature. Un enregistrement est un ensemble organisé de données.

* Enfin, les *OPERATEURS* qui sont les programmes mentionnés ci-dessus et correspondant à l'implémentation informatique des COMMANDES définies dans la phase fonctionnelle. Ces commandes seront analysées et puis implémentées pour engendrer un certain nombre de programmes. Les OPERATEURS sont mis en correspondance par les informations qu'ils manipulent avec la structure logique de données (R16).

1.c.2. Description fonctionnelle du modèle

Le modèle général étant décrit très sommairement, nous allons spécifier chacun de ses composants. En effet, les éléments des différents composants ont des propriétés qui leur sont propres, ils sont souvent en relation avec d'autres éléments et quelquefois ces relations ont elles aussi des propriétés caractéristiques.

Nous allons faire une description des composants par des tableaux. Un tableau permet de spécifier pour chaque composant les PROPRIETES de ses éléments. En général, c'est sous forme abrégée que l'on nommera les propriétés des éléments (IDENTIFICATEUR) ; c'est pourquoi, une brève description ou EXPLICATION s'impose. Si le composant est en RELATION AVEC D'AUTRES COMPOSANTS, on pourra citer l'IDENTIFICATEUR de cette relation avec son EXPLICATION et le nom du COMPOSANT EN RELATION. Enfin, on définira les PROPRIETES des relations.

REMARQUE : Certaines PROPRIETES sont volontairement assignées à tous les éléments de tous les composants. En effet, on dit que tout élément a un CODE, un LIBELLE et une DEFINITION :

- Le CODE est la propriété de tout élément qui lui permet de se différencier des autres éléments. Deux éléments différents, qu'ils appartiennent au même composant ou à des composants divers, doivent toujours se différencier par la valeur de leur CODE. Autrement dit, le CODE est une propriété discriminante de l'élément dans tout le système.

- Le LIBELLE d'un élément permet en quelques mots de rappeler la signification de la valeur du CODE. En effet, le code étant, dans la plupart des cas, une combinaison de chiffres et de lettres (contraction mnémonique), le LIBELLE permet de donner une signification à l'élément.

Exemple :

CODE d'un élément E = GSPINFO
LIBELLE de E = "Groupe de spécifications informatiques".

- La DEFINITION est une explication sous forme de texte libre en français, aussi complète que possible de l'élément.

DEFINITION de E = "Ce groupe doit fournir et contrôler les spécifications informatiques. Il sera constitué des ... etc".

Nous ferons par la suite la description des différents constituants du modèle.

1.c.2.1. Constituants permettant de décrire l'organisation où s'insère le projet informatique : les GROUPES FONCTIONNELS, les SERVICES ADMINISTRATIFS et les PERSONNES

COMPOSANT = GROUPES FONCTIONNELS

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEGF	Code du groupe fonctionnel					
LIBELGF	Libellé du groupe					
DEFGF	Définition du groupe					
		MEMBRESGF	PERSONNES	Liste des personnes qui sont rattachées au groupe fonctionnel		
		RESPONSABILITE	Tous les composants du projet	Liste de composants spécifiés par le groupe fonctionnel	PAR-QUI	La PERSONNE appartenant au groupe qui a donné les spécifications d'un composant

REMARQUES :

(1) Les GROUPES FONCTIONNELS sont les responsables de la conception de l'application. Ils sont constitués par les personnes qui participent aux spécifications du projet, c'est-à-dire des personnes qui utilisent MACSI-1. Pourquoi distinguer plusieurs groupes fonctionnels plutôt qu'un seul ?

Il est souhaitable de distinguer plusieurs groupes dans la mesure où les fonctions de chaque personne dans l'équipe de conception et de réalisation d'un projet informatique ne sont pas identiques. A titre d'exemple (car il n'existe pas une seule structure efficace de gestion d'un projet), il est possible de distinguer :

- Le chef du projet responsable en termes de délais et de coûts de l'application.
- Le groupe chargé des spécifications informatiques.
- Le groupe chargé du suivi technique de la réalisation informatique.

Chacune de ces fonctions doit faire l'objet de la déclaration d'un groupe fonctionnel (cf. Description de EQUIPE dans MACSI-1, au Chapitre 2 p. 2.6.) dès le début du projet.

(2) L'association entre les spécifications d'un composant et la personne qui les donne est intéressante uniquement si nous supposons, ce qui est le cas, qu'aucune autre personne ne peut les modifier. Nous contrôlerons par les programmes de diagnostic de PACS que les spécifications supplémentaires ou la modification d'un composant sont exécutées par cette même personne. Néanmoins, le responsable du projet pourra remplacer l'affectation d'une personne à un composant par une autre personne appartenant au même groupe.

COMPOSANT = SERVICES ADMINISTRATIFS

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODSER	Code identificateur du Service					
LIBELSER	Libellé du Service					
DEFSER	Définition du Service					
		EMPLOYES	PERSONNES	Liste de personnes appartenant aux Services Administratifs ayant une responsabilité dans le projet.		
ADRESER	Adresse du Service					
TELSER	Téléphone du Service					

REMARQUES :

(1) Un SERVICE ADMINISTRATIF est tout département, division, service, cellule ou poste de l'organisation concernée par le projet informatique. Pour qu'un Service Administratif soit concerné par l'application, il faut qu'au moins l'un de ses membres participe au projet en tant qu'utilisateur ou bien comme spécificateur, c'est-à-dire appartenant soit à un groupe d'exécution, soit à un groupe fonctionnel.

(2) Une personne est rattachée à un Service Administratif et à un seul (cf. REMARQUE 2 p. 1.23.).

(3) Ce composant pourrait être complété à la demande par une hiérarchisation des services ; mais rappelons que nous décrivons ici uniquement les principes généraux du modèle MACSI-1.

COMPOSANT = PERSONNES

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEPER	Code de la personne					
NOMPER	Nom de la personne					
PRENOMPER	Prénom de la personne					
ADRESPER	Adresse de la personne					
TELPER	Téléphone de la personne					
		ADMINISTRATION	SERVICES ADMINISTRATIFS	Service administratif auquel la personne appartient		
		ROLE	GROUPES FONCTIONNELS GROUPES D'EXECUTION	Liste des groupes fonctionnels et/ ou d'exécution auxquels la personne est rattachée	FONCTION	Description de la fonction de la personne dans le groupe.

REMARQUES :

(1) Une PERSONNE participe au projet en tant que membre d'un groupe fonctionnel ou d'un groupe d'exécution. La caractéristique FONCTION d'une personne dans un groupe, fonctionnel ou d'exécution, est différente ou plus précise que la fonction du groupe définie par DEFGF ou DEFGE.

(2) Toute Personne doit appartenir à un service administratif et à un seul. Ceci permet de l'identifier dans son rattachement administratif en particulier dans la manière dont elle est hiérarchiquement située et à quel titre elle est rémunérée. Le rattachement à un service administratif n'est en rien synonyme d'un rattachement géographique ou fonctionnel. Par exemple, un enseignant est affecté à une Unité d'Enseignement et de Recherche, il peut néanmoins assurer fonctionnellement son service dans une autre unité : être en mission ...

(3) Toute Personne doit appartenir au moins à un groupe,que celui-ci soit fonctionnel ou d'exécution. Si ce n'était pas le cas, figureraient dans le cadre du projet des personnes n'ayant aucun rôle. Par contre, une Personne peut très bien appartenir à plusieurs groupes d'exécution. Ceci signifie qu'elle partagera son activité au moment où le projet informatique fonctionnera entre plusieurs fonctions différentes. Par exemple, une Personne peut très bien être chargée de participer au groupe de contrôle des bordereaux d'entrée, et par ailleurs, être chargée de participer au groupe d'exécution responsable d'exploiter un document de sortie.
Une Personne peut aussi participer à un groupe fonctionnel et à un groupe d'exécution. Ceci signifie qu'au moment de la conception du projet elle a eu un rôle actif de spécification en tant que membre d'un groupe fonctionnel, mais qu'une fois le projet opérationnel, elle continue à participer à son fonctionnement en tant que membre d'un groupe d'exécution.

1.c.2.2. Constituants qui décrivent les opérations dans le projet informatique : Les GROUPES D'EXECUTION, les ACTIONS, les COMMANDES et les MODULES

COMPOSANT = GROUPES D'EXECUTION

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEGE	Code d'identification du groupe					
LIBELGE	Libellé du groupe d'exécution					
DEFGE	Définition du groupe					
		MEMBRESGE	PERSONNES	Liste de personnes rattachées au groupe d'exécution		
		COMMANDES-UTILISABLES	COMMANDES	Liste de toutes les commandes utilisables par le groupe d'exécution		
		ACTIONS-ASSUREES	ACTIONS	Liste de toutes les actions dont le groupe a la responsabilité d'exécution.		

REMARQUES :

(1) Un GROUPE D'EXECUTION est un ensemble de personnes qui auront la même responsabilité d'exécution lorsque le projet sera opérationnel. Il sera chargé de l'exécution des actions qui lui sont affectées, de même qu'il devra lancer l'exécution des programmes associés aux commandes ayant été spécifiées pour lui.

(2) Dans une première étape de spécification d'un projet, la déclaration de ces groupes est associée à celle des commandes et des actions dont ils seront chargés. La liste des personnes qui participeront au groupe d'exécution ne sera définie qu'au moment du lancement du projet, lorsqu'on mettra tous les moyens en place. Par exemple, au moment de la conception du projet, on dira qu'il existe un groupe d'exécution X chargé du remplissage du bordereau Y, mais on n'est pas capable de fixer tout de suite la liste des personnes qui appartiendront à ce groupe X. Donc, un groupe d'exécution est considéré comme fonctionnellement chargé de l'exécution d'un certain nombre d'actions et de commandes, ce n'est que plus tard, au niveau organique, que seront précisées les personnes qui doivent répondre à cette fonction en faisant partie du groupe.

COMPOSANT = COMMANDES (I)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODECO	Code de la commande					
LIBELCO	Libellé de la commande					
DEFCO	Définition de la commande					
TYPCO	Type de la commande (entrée, contrôle ou sortie)					
ETATCO	Représente l'état dans lequel se trouve la commande					
DATE-1-CO	Permet d'enregistrer la date de prise en compte de la commande par le responsable informatique					
DATE-2-CO	Date prévue par le responsable informatique de fin d'analyse et programmation de la commande par l'analyste-programmeur					
DATE-3-CO	Date de réception par le responsable informatique de la commande implémentée sous forme d'opérateur					

COMPOSANT = COMMANDES (II)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
DATE-4-CO	Date de réception par le demandeur (groupe fonctionnel) de la commande sous forme d'opérateur					
MODIFCO	Liste des modifications subies par la commande. On retrouvera pour chaque modification :					
DATE-1-MO DATE-2-MO DATE-3-MO DATE-4-MO	Même signification que les dates précédentes					
DATE-5-MO	Date d'annulation de la commande					
		DEMANDEUR	PERSONNES	Personne du groupe fonctionnel demandeur de la commande		
		EXECUTANTS-CO	GROUPES D'EXECUTION	Liste des groupes d'exécution qui peuvent ou qui doivent, à un certain moment, activer la commande		
		DOC-ASSOCIE	DOCUMENTS	Liste des documents en entrée ou en sortie associés à la commande		

COMPOSANT = COMMANDES (III)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
		OPERATEUR-ASSOCIE	OPERATEURS	Opérateur exprimant la logique de la commande		

REMARQUES :

(1) Une COMMANDE correspond à la demande d'un groupe fonctionnel d'un traitement automatisé de l'information. La personne qui spécifie une Commande donne dans DEFCO une définition aussi complète que possible de celle-ci sous forme d'un texte libre en français. Cette personne est en général sans compétence particulière en informatique. Elle devra faire référence aux différentes "données" manipulées par cette commande :

- s'il s'agit d'une commande dont le type est entrée-contrôle, elle fera référence aux différentes informations stockées sur les documents associés à cette Commande.
- s'il s'agit d'une Commande de type sortie, elle devra faire référence aux différentes données qui doit produire la Commande et celles dont elle a besoin.

(2) A un moment donné de l'analyse du projet, une Commande peut être dans l'un des états suivants :

1 - En attente d'examen par l'équipe informatique : la Commande a été spécifiée et enregistrée, mais le responsable informatique n'a pas encore chargé d'analyste-programmeur de la définition de sa logique.

2 - En attente d'informations complémentaires : il n'y a, en effet, pas d'accord réciproque entre le demandeur et l'analyste-informaticien qui examine le contenu de la définition spécifiée par le demandeur. En général, les spécifications seront trop vagues et l'analyste informaticien demande des précisions complémentaires. Une fois qu'il les aura obtenues, la définition sera évidemment modifiée.

3 - En attente de modification de la structure des données : en effet, les utilisateurs peuvent demander la réalisation de Commandes qui sont impossibles soit à cause de la complexité algorithmique des traitements qu'elles supposent, soit à cause des structures de fichiers et de base de données retenues. Ces Commandes sont provisoirement abandonnées et seront peut-être implémentées dans une seconde version de l'application. Les Commandes en état 3 mesurent, en quelque sorte, l'écart qui existe entre la version opérationnelle d'un projet et les souhaits des utilisateurs.

4 - La Commande est en cours d'implémentation.

5 - La Commande implémentée est réceptionnée, après jeu d'essai, par le responsable informatique.

6 - La Commande implémentée est réceptionnée par le demandeur.

La figure 1-6 nous montre le graphe de passage des états :

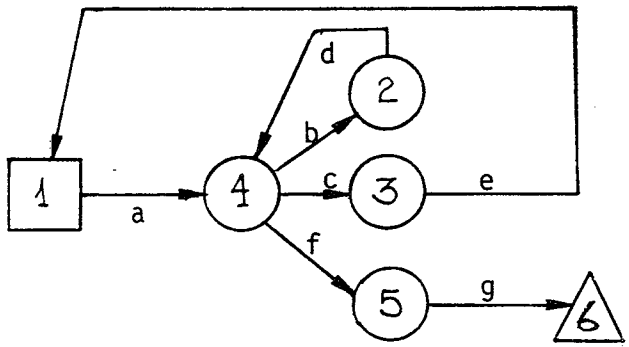


Figure 1-6

La Commande est au départ dans l'état [1] . Une fois qu'elle est examinée par le responsable informatique (DATE-1-CO), elle est l'objet d'une analyse par l'analyste programmeur responsable de l'implémentation de la Commande : elle passe dans l'état (4) . A la fin de cette phase d'analyse (DATE-2-CO), plusieurs situations peuvent se présenter :

a) La Commande passe dans l'état (2) si l'analyste-programmeur n'a pas les précisions suffisantes.

b) La Commande passe dans l'état (3) si l'analyste-programmeur pense ne pas pouvoir l'implémenter.

c) L'analyste-programmeur fait la programmation de la Commande et la livre au responsable informatique (DATE-3-CO). La Commande est dans l'état (5) . Quand le responsable informatique livre la Commande à l'utilisateur (DATE-4-CO), la Commande passe dans l'état △6 .
Si la Commande est dans l'état (2) les informations supplémentaires données par l'utilisateur feront l'objet d'une analyse et programmation. Elle passe à l'état (4) .
La Commande qui est dans l'état (3) ne peut que passer à l'état initial [1] lorsque des modifications importantes auront été faites.

Résumons les sept fonctions (arcs du graphe) de changement d'état d'une commande :

a : Prise en compte de la commande.

b : L'analyste déclare qu'il a besoin de précisions supplémentaires sur la définition de cette commande.

c : Implémentation de cette commande actuellement impossible.

d : La définition de la commande a été complétée.

e : Le responsable du projet décide que la commande peut être de nouveau implémentée.

f : Le responsable du projet a réceptionné la programmation de la commande.

g : La commande est réceptionnée par le demandeur.

(3) Une même Commande peut être associée à plusieurs groupes d'exécution. Ceci veut dire qu'une même logique de traitement automatisé de l'information peut être applicable à différents "endroits" et à différents moments du projet. Par exemple, un même document récapitulatif peut être demandé par la direction (groupe d'exécution) ou par le comptable (groupe d'exécution) à deux moment différents.

(4) Le spécificateur aura l'opportunité de nommer les documents qui sont associés en entrée ou en sortie de la Commande. Ce ne sera que plus tard que ces documents seront définis.

COMPOSANT = ACTIONS

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEAC	Code identificateur de l'action					
LIBELAC	Libellé de l'action					
TYPAC	Type de l'action					
ETATAC	Etat de l'action					
RESULTAT	Description des résultats attendus par l'exécution de l'action					
		EXECUTANTS-AC	GROUPES D'EXECUTION	Liste de groupes d'exécution qui sont responsables d'accomplir la tâche exprimée par l'action		
		SPECIFICATEUR	PERSONNES	Personne qui a spécifié l'action		

REMARQUES :

(1) Une ACTION est toute opération qui n'est pas une commande de traitement de l'information automatisé dans le cadre du projet et dont l'existence est indispensable pour que le projet fonctionne. L'ensemble des actions doit permettre de décrire tout l'environnement en amont et en aval des procédures informatiques dans lesquelles celles-ci sont insérées et prennent leur finalité. Autrement dit, les Actions décrivent l'organisation pour laquelle est mise en place l'application informatique. Cependant, cette première approche de la notion d'Action est insuffisante pour permettre une description de l'organisation environnant une application informatique.

Pour affiner la définition d'une Action, une première possibilité consiste à définir une classification qui apporte une différenciation, admise par tous, entre les différents types d'Actions. Cette classification doit avoir une double qualité :

- être sans ambiguïté, c'est-à-dire qu'une Action déterminée doit être classée sans hésitation par n'importe quel analyste dans une rubrique et une seule de cette classification.
- être pertinente pour la conduite du projet et permettre des diagnostics intéressants sur sa conception.

A titre indicatif, nous proposons la classification donnée par F. PECCOUD (réf. [22]) pour distinguer les différentes actions :

TYPE = 1 Action Décision
- 11 = Décision concernant le long terme (plus de 5 ans)
- 12 = Décision concernant le moyen terme (de 2 à 5 ans)
- 13 = Décision concernant le court terme (dans le cadre de l'année d'exercice en cours).

TYPE = 2 Action de production, de fabrication
- 21 = Conception d'un produit ou d'un service
- 22 = Fabrication du produit ou exécution du service
- 23 = Contrôle de qualité du produit.

TYPE = 3 Contrôle d'exécution
- 31 = Contrôle financier
- 32 = Contrôle de délai de la production
- 33 = Contrôle d'activité des individus.

TYPE = 4 Action de traitement de l'information non automatisée

41 = Création ou mise à jour d'un document

42 = Duplication d'un document

43 = Transcodification d'un document

44 = Transmission d'un document

45 = Contrôle manuel du contenu d'un document.

(2) L'ETAT d'une Action à un moment donné de l'analyse peut être l'un des suivants :

1 = L'Action est fonctionnellement définie.

2 = Il existe un manuel utilisateur associé à l'Action définie.

3 = L'Action a été simulée dans son exécution.

4 = L'Action est approuvée après simulation et déclarée opérationnelle.

(3) La définition de l'Action recouvre les spécifications des conditions d'exécution de cette action, y compris le rôle du groupe d'exécution. Cette définition est faite sous forme de texte libre en français.

COMPOSANT = MODULES (I)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEMO	Code identificateur du module					
LIBELMO	Libellé du module					
DEFMO	Définition du module					
		ELEMENTS	ACTIONS COMMANDES MODULES	Liste des éléments (actions, commandes ou modules) composant le module	EXECUTANT-EL	Groupe d'exécution associé à l'élément (action ou commande)
					SUITE	Liste des éléments (actions, commandes et/ou modules), qui, dynamiquement, s'exécuteront après l'élément selon certaines conditions. Chaque élément dans la liste a les propriétés suivantes : ELEMENT-SUITE = code identificateur de l'élément suivant.

COMPOSANT = MODULES (II)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
						CONDITION-SUITE = pour exprimer les conditions d'exécution autres que les conditions de type temporelles ou séquentielles QUAND = pour exprimer les conditions de type temporelles.
					PRECEDE	Liste des éléments (actions, commandes et/ou modules) qui s'exécutent immédiatement avant l'élément.
		ANALYSTE	PERSONNES	Personne du groupe fonctionnel qui a spécifié le module		

REMARQUES :

(1) Un MODULE est une structure composée de commandes, d'actions et même de modules appelés ses ELEMENTS. La notion de Module dans MACSI-1 doit permettre de représenter l'enchaînement dynamique des commandes et des actions.

Les règles de composition entre les éléments d'un Module sont les suivantes :

a) Séquence simple : Si un élément A est suivi d'un élément B, B sera déclaré comme un élément suite (ELEMENT-SUITE) de A.

b) Choix simple : Un élément A est suivi de B si la condition CO-1 est vérifiée, ou de C si la condition CO-2 l'est. B et C seront deux éléments de SUITE de A, chacun ayant dans CONDITION-SUITE et QUAND les textes définissant CO-1 et CO-2. Comme CO-1 et CO-2 s'expriment par des textes, il faut noter que CO-2 n'est pas forcément la négation de CO-1.

c) Processus indépendants : Un élément A est suivi de B et de C qui s'exécuteront en parallèle indépendamment l'un de l'autre. Alors A aura deux éléments dans sa liste SUITE, mais sans conditions d'exécution pour chaque élément.
Plus généralement, un élément A peut avoir n éléments dans la liste SUITE, $S_1 \ldots S_i \ldots S_n$. Quand un élément S_i est conditionné par une condition CO_i (dans CONDITION-SUITE et/ou QUAND), il exprime dans quelle situation S_i peut être activé simultanément avec $S_1 \ldots S_n$.

(2) Pour chaque élément (commande, action ou module), on dira quels sont leurs éléments-suite et les conditions d'exécution. Pour exprimer les conditions sous forme de texte, on a les caractéristiques CONDITION-SUITE et QUAND. Au moment de spécifier une condition temporelle (QUAND) ou non (CONDITION-SUITE), trois situations différentes peuvent se présenter :

a) L'analyste ou spécificateur connaît les conditions d'exécution qu'il exprimera sous forme de texte libre dans CONDITION-SUITE et/ou QUAND.

b) L'analyste sait qu'il n'y a pas de conditions particulières pour l'exécution de l'élément suivant. Autrement dit, l'exécution de l'élément suivant sera déclanché dès que l'exécution de l'élément précédent sera finie.

c) L'analyste ne connaît pas précisément les conditions d'exécution de l'élément suivant. Il veut reporter ultérieurement la spécification de ces conditions.

(3) L'analyste qui spécifie un Module doit être aussi celui qui spécifiera ensuite dans le détail les propriétés des actions et commandes constituant le Module.

(4) Les Modules permettent aussi la spécification des groupes d'exécution associés aux commandes et aux actions. On pourra dire quel groupe d'exécution est responsable de l'exécution d'une action, ou quel groupe d'exécution est chargé du lancement d'un programme. Un cas particulier se présente : Une commande peut être un programme qui doit s'exécuter sans l'intervention d'un groupe d'exécution, mais de façon automatique lorsque le moment précis sera arrivé. Par exemple, pour l'édition mensuelle des bulletins de paie, il est inutile chaque mois qu'un groupe d'exécution active le programme d'édition, celui-ci sera automatiquement exécuté tous les premiers du mois.

Prenons un exemple simple d'application informatique pour la décrire suivant une structure de Module. Ceci nous permettra de mieux visualiser les remarques (1) et (2) et d'en dégager d'autres.
On va décrire le Module concernant la gestion trimestrielle des examens dans un établissement scolaire. On appellera ce Module M10.

MODULE	ELEMENTS			SUITE		
CODEMO	CODEL	Brève description de l'élément	EXECUTANT-EL	ELEMENT-SUITE	CONDITION-SUITE	QUAND
M10	M11 (ELEMENT D'ENTREE)	Elaboration du planning des examens		A12		15 jours avant le commencement des examens
				A13		
	A12	Affichage du planning sur les panneaux	Service Scolarité	M14		
	A13	Préparation des sujets des examens	Groupe de professeurs	A15		4 jours avant le commencement des examens
	A15	Tirage des sujets d'examen	Service Tirage	M14		
	M14	Déroulement des examens		A16		
	A16	Correction des examens	Groupe de professeurs	A17		15 jours après le dernier examen
	A17	Perforation des résultats	Equipe de perforation	C18		
	C18	Contrôles directs et indirects sur les élèves et les notes	Equipe d'exploitation	M19	S'il y a détection d'erreurs par la commande	
				C20	Si pas d'erreurs	
	M19	Correction d'erreurs		A17		
	C20	Actualisation du fichier des NOTES	Equipe d'exploitation	C21		
				C25		
	C21	Edition des NOTES par groupe, par étudiant et par matière	Equipe d'exploitation	SORTIE		
	C25	Edition des résultats par moyenne décroissante	Equipe d'exploitation	SORTIE		

MODULE	ELEMENTS			SUITE		
CODEMO	CODEL	Brève description de l'élément	EXECUTANT-EL	ELEMENT-SUITE	CONDITION-SUITE	QUAND
M11	A22 (ELEMENT ENTREE)	Communication du planning prévisionnel aux professeurs	Service Scolarité	A23		
	A23	Communication des accords ou des critiques auprès du Service Scolarité	Groupe de professeurs	A24	S'il existe des critiques ou des désaccords sur le planning	
				SORTIE	Pas de désaccords	
	A24	Réajustement du programme en fonction des désiderata	Service de Scolarité	A22		

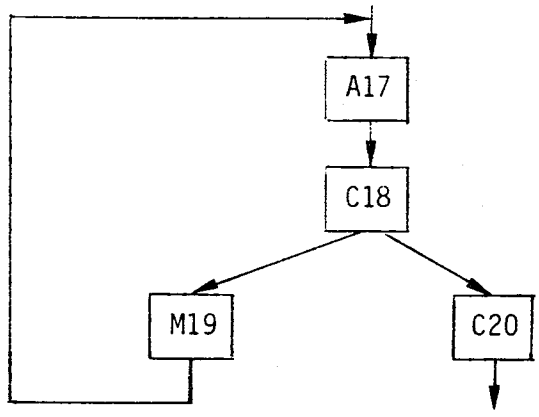
CONVENTION = Les Codes commençant par M sont des Codes de Modules
Les Codes commençant par A sont des Codes d'Actions
Les Codes commençant par C sont des Codes de Commandes.

(5) Chaque Module n'a qu'une seule entrée (l'élément marqué avec ELEMENT D'ENTREE) et une ou plusieurs sorties.
Exemple : L'entrée de M10 est M11 et ses sorties sont C21 et C25
L'entrée de M11 est A22 et sa sortie est A23.

(6) Un Module étant une combinaison d'actions, commandes et modules différents, il est évident qu'une affectation d'un groupe exécutant n'est pas possible au niveau du module (mais implicitement ce module concernera tous les groupes d'exécution affectés à un élément de ce module).

(7) Pour les éléments suite qui n'ont pas de conditions particulières d'exécution, on a pris l'option de laisser en blanc les rubriques CONDITION-SUITE et QUAND. On pourrait prendre (ce que l'on fera plus tard) une autre option plus explicite en utilisant un mot-clé, par exemple, RAS pour Rien A Signaler.

(8) La notion de boucle est présente dans la description de la structure des Modules. En effet, dans la description du Module M10 au niveau de l'action A17, on a la structure graphique suivante :



L'élément suivant du Module M19 est l'action A17 qui a été déclaré antérieurement. On a la boucle A17 - C18 - M19 - A17.
Un autre exemple peut se dégager de la description du Module M11. En effet, la série A22-A23-A24-A22 est aussi une boucle.

Un programme de PACS vérifiera qu'il y a toujours une possibilité de sortie d'une boucle (par exemple, la commande C20 est une possibilité de sortir de la boucle A17 - C18 - M19).

(9) Dans un Module, un élément correspond à l'exécution d'un processus, que celui-ci soit une action, une commande ou un module. L'activation de ce processus peut être, dans l'ensemble de l'application et même d'un module, demandée plusieurs fois à la suite d'éléments différents. La relation entre éléments d'un module est donc ici plutôt analogue à celle existant entre les instructions d'un programme.

(10) L'analyse d'un projet sous forme de Modules oblige une analyse descendante de celui-ci, c'est-à-dire une construction du projet en posant d'abord les fonctions macroscopiques (les Modules) affinées de plus en plus jusqu'à leurs détails les plus fins (les actions et les commandes).

(11) Si le lecteur est informaticien, il peut constater que les spécifications de Modules pourraient être plus rigoureuses qu'elles ne le sont, en particulier au niveau des conditions des suites d'un élément et sur la manière d'exprimer l'existence de processus indépendants. Vous trouverez dans [22], [14], [17] une formalisation d'un langage de spécification de modules. Mais, dans MACSI-1, un des objectifs est que les groupes responsables de la spécification générale de l'application ne soient pas constitués uniquement d'informaticiens mais de spécialistes de gestion capables, avec une formation technique très limitée en informatique, de fournir la structure générale du projet qu'ils demandent. Le niveau de complexité du concept de Module dans MACSI-1 nous semble être le maximum de ce qui peut être demandé à quelqu'un n'ayant pas une formation en informatique.

(12) Dans les dossiers classiques d'analyse fonctionnelle, on trouve des diagrammes de circulation de l'information qui montrent les différentes étapes des transformations des données par rapport aux services administratifs dans lesquels ils circulent. Néanmoins, ces diagrammes présentent des inconvénients car ils expriment rarement les conditions de passage entre les différentes étapes. Aussi, les postes de transformation de l'information sont les services administratifs de l'organisation, lesquels, comme on l'a vu précédemment, n'ont pas une signification fonctionnelle par rapport à l'application.

1.c.2.3. Constituants permettant la description des données concernées par l'application = les DOCUMENTS, les ENSEMBLES, les RELATIONS et les PROPRIETES.

A - Les Documents

COMPOSANT = DOCUMENTS

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEDO	Code d'identification du document					
LIBELDO	Libellé du document					
DEFDO	Définition du document					
TYPDO	Type du document					
		CONTENU	ENREGISTREMENTS	Enregistrement associé au document		
		CLEDO	PROPRIETES	Propriété servant comme critère de tri du document		
		FICHIERS-ASSOCIES	FICHIERS	Liste de tous les fichiers qui sont déduits du document ou originaires du document		

REMARQUES :

(1) Les DOCUMENTS sont les supports de l'information directement lisibles par les personnes appartenant aux groupes d'exécution. Ils contiendront des informations qui seront traitées par une commande, ou qui seront les sorties d'une commande. On considère comme Documents les listings provenant des imprimantes (qu'ils soient ou non en partie préimprimés), les sorties sur écran cathodique, les entrées ou sorties par téléimprimeur, les bordereaux de perforation.

(2) Les données qui figureront sur un Document seront répertoriées dans CONTENU. Le contenu nous permettra d'énumérer toutes les données figurant sur le document. Ces données seront indiquées sous forme de PROPRIETES (le composant PROPRIETE sera défini à la fin du paragraphe 1.c.2.3.). Ces propriétés seront regroupées pour former un ENREGISTREMENT (définition au paragraphe 1.c.3.1.). La définition du document (DEFDO) nous permet de faire une description de la disposition des données sur le document (mise en page, présentation externe, massicotage, reliure, ...).

(3) Un Document peut être de deux types (TYPDO) :

- . Document d'entrée d'une commande
- . Document en sortie d'une commande.

B - Structure logique des Données

B-1 - Les spécifications fonctionnelles d'une application informatique ne sont pas complètes si la structure des données n'est pas spécifiée. Or, nous constatons qu'il n'existe aucune homogénéité dans les formalismes retenus par les différentes méthodes d'analyse pour la description des données et des traitements. Ce n'est pas seulement un problème de vocabulaire, mais ces méthodes, ainsi que les différentes études théoriques, diffèrent aussi dans les modèles fondamentaux de représentation sémantique. Nous renvoyons le lecteur aux travaux de J.R. ABRIAL ([1], [2]), de C. DELOBEL ([11]) et de F. PECCOUD ([23], [26]) pour approfondir ce problème.

B-2 - Les différents objectifs retenus pour définir le formalisme de description des données dans MACSI-1 sont :

a) La sémantique du problème doit être clairement et aussi totalement que possible exprimée avant que les choix d'implémentation soient fixés.

b) Le modèle sémantique des données doit être compréhensible par les utilisateurs qui participent aux spécifications du projet. Il doit donc ne pas proposer une terminologie trop compliquée et trop spécialisée.

c) Le formalisme d'expression de la structure logique de données doit permettre de représenter à la fois des fichiers et des bases de données, puisque nous avons supposé que, dans les projets spécifiés et documentés avec MACSI-1 (cf. § 1.c.1.) ces deux types d'outils de gestion des données peuvent être mis en oeuvre.

B-3 - <u>Le modèle sémantique retenu pour la structure logique de données</u> :

Les données dans une application informatique sont une image d'un univers physique dans lequel nous distinguons plusieurs composants : les ENSEMBLES, les PROPRIETES de ces ensembles et des RELATIONS entre ENSEMBLES.

COMPOSANT = ENSEMBLES DE BASE

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODENS	Code identificateur de l'ensemble					
LIBELENS	Libellé de l'ensemble					
DEFENS	Définition de l'ensemble					
TAILLENS	Taille (cardinal) de l'ensemble					
		PROPRIETES-ENS	PROPRIETES	Liste des propriétés des éléments de l'ensemble		
		CLEFS-ENS	PROPRIETES	Sous-ensemble de PROPRIETES-ENS. Ces propriétés sont considérées comme critère de tri ou comme index de l'ensemble		

REMARQUES :

(1) Les ENSEMBLES de BASE sont des ensembles d'éléments concrets (produits, machines, locaux, personnes, etc ...) dont l'existence est évidente dans le système physique environnant le projet informatique.

Exemple : Prenons comme illustration des notions présentées l'application de gestion des examens dans un établissement scolaire ; on peut dans ce cas distinguer comme ensembles :

E_1 = Les Professeurs
E_2 = Les salles de cours
E_3 = Les matières
E_4 = Les examens
etc ...

(2) Chaque élément d'un Ensemble de base est particularisé par les valeurs de ses propriétés (PROPRIETES-ENS), et il est identifié par la valeur d'une de ces propriétés qui est discriminante, généralement appelée "code".

Exemple :

Ensembles de base	Propriétés de ces Ensembles (PROPRIETES-ENS)
E_1 = Les Professeurs	P_{11} = Nom P_{12} = Prénoms P_{13} = Catégorie P_{14}=Etat-Civil
E_2 = Les Salles de cours	P_{21} = Numéro P_{22} = Capacité P_{23} = Coordonnées
E_3 = Les Matières	P_{31} = Titre P_{32} = Coefficient P_{33} = Niveau d'enseignement

(3) La taille d'un Ensemble de base (TAILLENS) permet d'affiner la signification de l'Ensemble. En effet, on donne une borne supérieure du cardinal de cet Ensemble durant toute la période pendant laquelle on identifiera les éléments de cet Ensemble.

Exemple : L'Ensemble de professeurs peut varier pendant la période où on gèrera cet établissement. Néanmoins, on peut affirmer qu'il n'excédera pas la centaine des professeurs. On dira que E_1 a une taille de 100.

COMPOSANT = ENSEMBLES-RELATION (I)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODER	Code d'identification de l'ensemble-relation					
LIBELER	Libellé de l'ensemble-relation					
DEFER	Définition de l'ensemble-relation					
TAILLER	Taille de l'ensemble-relation					
		ARG-1	ENSEMBLES ENSEMBLES-RELATION	Premier argument de la relation	ACCES-1	Fonction de ARG-1 dans $\mathcal{P}$(ARG-2)
					MIN-1	Cardinaux minima et maxima des sous-ensembles définis par la fonction d'accès ACCES-1
					MAX-1	
		ARG-2	ENSEMBLES ENSEMBLES-RELATION	Deuxième argument de la relation	ACCES-2	Fonction de ARG-2 dans $\mathcal{P}$(ARG-1)
					MIN-2	Cardinaux minima et maxima des sous-ensembles définis par la fonction d'accès ACCES-2
					MAX-2	

COMPOSANT = ENSEMBLES RELATION (II)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
		PROPRIETES-REL	PROPRIETES	Liste des propriétés de l'ensemble-relation		
		CLEFS-REL	PROPRIETES	Sous-ensemble de PROPRIETES-REL. Ces propriétés sont considérées comme critères de tri, d'index, de l'ensemble-relation.		

REMARQUES :

(1) Il existe dans les applications informatiques des RELATIONS BINAIRES entre les ensembles. Chaque relation est définie comme une partie du produit cartésien de deux ensembles, E_i et E_j appelés les ARGUMENTS de la RELATION.

Exemple :

RELATION	LIBELLE	ARGUMENT-1	ARGUMENT-2
R1 = ENSEIGNEMENT	Activités pédagogiques des professeurs	E_1 = PROFESSEURS	E_3 = MATIERES
R2 = COURS	Planning professeurs-matières-salles	R1 = ENSEIGNEMENT	E_2 = SALLES DE COURS

Une relation binaire $R_k \in E_i \times E_j$ est donc aussi un ENSEMBLE formé par les couples $r \equiv (\ell_{ip}, \ell_{jq})$ où $\ell_{ip} \in E_i$ et $\ell_{jq} \in E_j$. On appelle ces ensembles les ENSEMBLES-RELATION.

(2) Chaque relation elle-même peut, en tant qu'ensemble, avoir des propriétés (PROPRIETES-REL) qui lui sont spécifiques.

Exemple :

RELATION	PROPRIETE
R2	P200 = Durée de l'enseignement d'un professeur dans une salle.

(3) Pour enrichir la signification des ensembles-relations, on peut définir pour une relation R entre les ensembles E_i et E_j (ses arguments) deux fonctions d'accès. La fonction d'accès f_i de l'ensemble E_i vers l'ensemble E_j est le nom que l'on donne au sous-ensemble de E_j, étant en relation R avec un élément e_i de E_i (cf. J.R. ABRIAL [1]).

Exemple :

Ensemble E_1 = Professeurs Ensemble E_2 = Matières Relation R = Enseignements

Fonction d'accès f1 = "Spécialités" (de un professeur) est le nom d'un sous-ensemble de E_2 (Matières) dont les éléments sont reliés au professeur par la relation R.

Fonction d'accès f2 = "Enseignants" (de une matière) est le nom d'un sous-ensemble de E_1 (Professeurs) dont les éléments sont reliés à la matière par la relation R.

Il est fort conseillé de donner trois noms différents à la relation et aux deux fonctions d'accès. Pour le nom de la relation, on pourra par exemple faire une contraction discriminante significative d'ARG-1 et ARG-2.

Exemple :

Ensemble E_1 = Elèves Ensemble E_2 = Examens

Ensemble-relation R_3 = Elève-exam ACCES-1 = "épreuves"
ACCES-2 = "inscrits"

(4) On peut définir les cardinaux maxima et minima du sous-ensemble défini par une fonction d'accès (cf. J.R. ABRIAL [1]).
Le cardinal maximum d'ACCES-1 s'appelle MAX-1.
Le cardinal minimum d'ACCES-1 s'appelle MIN-1.

Exemple :

Ensemble E_1 = Professeurs Ensemble E_2 = Matières Relation R1=Enseignements

ACCES-1 = "Spécialités" ACCES-2 = "Enseignants"

MIN-1 définit le minimum de matières qui peuvent être les "spécialités" d'un professeur.

Donc, dire que MIN-1 = 1, c'est dire qu'un professeur doit être compétent au moins dans une matière ;

dire que MAX-1 = 2, c'est dire qu'un professeur ne peut être capable d'assurer l'enseignement de plus de 2 matières ;

dire que MIN-2 = 1, c'est dire qu'une matière doit être enseignée au moins par un professeur ;

dire que MAX-2 = 4, c'est dire qu'une matière ne peut pas être enseignée par plus de 4 professeurs différents.

COMPOSANT = PROPRIETES (I)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEPP	Code d'identification de la propriété					
LIBELPP	Libellé de la propriété					
DEFPP	Définition de la propriété					
TYPP	Type de valeur de la propriété					
CONTROLE	Description des différents contrôles éventuels à effectuer sur la propriété (existence, cohérence, plage de valeurs, nature, etc ...)					
		DEQUOI	ENSEMBLE ENSEMBLE-RELATION	Dénote l'ensemble ou l'ensemble-relation auquel cette propriété appartient		
		COMPOSANTS-PP	PROPRIETES	Liste des propriétés composantes d'une propriété lorsqu'elle est composée		

COMPOSANT = PROPRIETES (II)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
		COMP-DE	PROPRIETES	Liste des propriétés composées dont celle-ci fait partie en tant que composante.		

REMARQUES :

(1) Les PROPRIETES des éléments d'un ensemble (par abus de langage, nous dirons : "PROPRIETES d'un ensemble") sont les caractéristiques fonctionnelles de ces éléments. Autrement dit, les PROPRIETES matérialisent ce qui est propre aux éléments d'un ensemble. L'identification de chaque élément d'un ensemble se fait en évaluant les VALEURS de ses Propriétés : deux éléments distincts diffèrent par au moins une valeur.

(2) Une Propriété peut être composée. En effet, si P_1 et P_2 sont des Propriétés des éléments d'un même ensemble, on peut définir une propriété P_3 composée de P_1 et P_2 dont chaque valeur, pour un élément donné, est la concaténation de P_1 à celle de P_2 pour cet élément.

Exemple :

Tous les éléments de l'ensemble "Salles de cours" ont une Propriété composée (P_4) qui est "COORDONNEES". Elle est composée de "nom-du-bâtiment" (P_1 = A,B,C ...), de "étage" (P_2 = 0,1,2 ...) et de "numéro" P_3 = 01,02, ...).

(3) Le type de valeur (TYPP) d'une Propriété peut être, selon le cas :

- TEXTE = Un texte est une valeur dont la seule utilité est de transmettre des messages et sert uniquement à l'édition. On ne pourra faire aucune recherche à partir de son contenu. Il n'existe aucune opération de traitement automatique.
  Exemple : "Ce texte ne peut être qu'édité".
- MOT = Ce sont des chaînes de caractères permettant d'effectuer une recherche par test d'égalité.
  Exemples : "Dupont", "324", "A15".
- LISTE DE VALEURS ALPHANUMERIQUES = On associe une liste de MOTS à la Propriété. Les valeurs que prendra la propriété au cours de son existence doivent toujours être l'une des valeurs de la liste associée.
  Exemple : L' "état-civil" d'un professeur doit être :
  - célibataire
  - marié
  - divorcé
  - veuf.

- NUMERIQUE ENTIER = Chaîne de caractères numériques pouvant faire l'objet d'opérations arithmétiques et de comparaisons par rapport à la relation d'ordre existant entre les nombres entiers.
- NUMERIQUE REEL = Les valeurs peuvent faire l'objet d'opérations arithmétiques et de comparaison par rapport à la relation d'ordre existant sur les nombres réels.
- LISTE DE VALEURS NUMERIQUE ENTIER = Tout élément de la liste de valeurs est de type NUMERIQUE ENTIER.
- LISTE DE VALEURS NUMERIQUE REEL = Tout élément de la liste de valeurs doit être de type NUMERIQUE REEL.

(4) Une Propriété pour un élément défini d'un ensemble a toujours une valeur et une seule. Nous verrons l'intérêt de cette contrainte ci-dessous (cf. § C.2.).

(5) Toute Propriété, quelle soit composante ou composée, ne peut appartenir qu'à un seul ensemble (ici, Propriété est à prendre au sens d'identificateur de propriété).

(6) Une Propriété peut être la composante d'une ou plusieurs propriétés composées d'un même ensemble.

(7) La définition d'une Propriété peut permettre d'ajouter des spécifications supplémentaires sur elle-même. Par exemple, si la propriété est du type liste de valeurs, on pourra spécifier cette liste, si la propriété est du type numérique, on pourra donner des bornes supérieure et inférieure de sa valeur, etc ....

C - Réflexion méthodologique : quelques points délicats dans la description sémantique des données

C-1 - Existe-t-il une différence de nature entre les PROPRIETES et les RELATIONS ? En effet, on pourrait définir le NOM d'un PROFESSEUR selon deux modèles différents :

MODELE 1 Ensemble E_1 = PROFESSEURS Propriété P_{11} = NOM (définie spécifiquement comme une propriété unique des éléments de l'ensemble E_1)

MODELE 2 Ensemble E_1 = PROFESSEURS Ensemble E_{10} = NOMS
Relation R_{11} ayant pour arguments E_1 et E_{10} associant à chaque PROFESSEUR un NOM.

Il existe pour nous une différence entre ces deux modèles :

* Dans le MODELE 1, la suppression de l'ensemble E_1 implique celle de toutes ses propriétés, et en particulier celle de P_{11}, car P_{11} est une propriété de E_1 et de E_1 seul.

* Dans le MODELE 2, par contre, la suppression de E_1 implique la suppression de R_{11} puisqu'un de ses arguments disparaît, mais l'ensemble E_{10} posé en tant que tel peut subsister, d'autant plus que E_{10} peut être argument d'autres relations.
Dans l'approche ensembliste de représentation des données que nous proposons, ce problème est résolu :

  - Une PROPRIETE est une caractéristique associée exclusivement à tous les éléments d'un ensemble. Si, après avoir déclaré son existence, cet ensemble est jugé inadéquat et sa non-existence est affirmée, sont supprimées implicitement toutes les propriétés qui lui étaient associées.

  - Par contre, une RELATION R entre deux ensembles E et E' ne peut être établie que si l'on peut répondre affirmativement à ces deux questions :

    . Existe-t-il d'autres relations de E' avec d'autres ensembles que E ?

    . Sa suppression de la relation R entre E et E' conserve-t-elle encore un sens à l'existence de E' indépendamment de E ?

  En général, la réponse négative à l'une de ces deux questions conduit à remplacer l'existence de E' et de R par la déclaration d'une propriété P' de E.

C-2 - Lorsqu'il existe une relation R entre deux ensembles E et E', il est souvent tentant de considérer les propriétés des ensembles comme étant aussi celles de la relation et réciproquement. Nous voyons alors l'intérêt de la règle imposant l'existence pour une propriété d'une valeur et une seule pour chaque élément d'un ensemble.

C-3 - Il n'est pas toujours évident de définir la fonction caractéristique d'un ensemble d'éléments pris dans l'univers réel uniquement en définissant ces propriétés. Des ambiguïtés peuvent subsister, même en convenant que tout élément de l'ensemble doit avoir une valeur définie pour toutes les propriétés associées à l'ensemble.

On peut déjà lever en partie cette difficulté en associant à l'ensemble une définition (DEFENS, DEFER) précise sous forme de texte libre, où chaque mot est significatif. On dira qu'on précise l'ensemble en compréhension. Mais ce n'est pas toujours possible.
Quelle définition donnera-t-on à l'ensemble E_1 des professeurs ?

E_1 Est-ce "les personnes fonctionnaires de l'Education Nationale ?"

E'_1 Est-ce "les personnes ayant une compétence culturelle et pédagogique dans une ou plusieurs matières scientifiques ou littéraires ?'

E''_1 Est-ce "les personnes professant soit dans l'enseignement primaire, soit dans l'enseignement secondaire, soit dans l'enseignement supérieur ?"

E'''_1 Est-ce "les personnes ayant les diplômes requis pour enseigner ?"

A défaut d'une définition correcte d'un ensemble en compréhension, il faut le définir en extension.

Ainsi, nous définissons E_1^* comme l'ensemble de personnes qui sont identifiées dans le fichier des personnes "enseignants" du Ministère de l'Education Nationale. E_1^* est-il un ensemble identique à E_1, E'_1, E''_1 ou E'''_1 ? Seule, une connaissance complète de la procédure de création et de mise à jour du fichier permet de répondre à la question.

1.c.3. Description organique du modèle

Les constituants décrits ici appartiennent aux spécifications organiques de l'application, c'est-à-dire qu'elles ne seront prises en compte dans l'analyse qu'à partir du moment où les constituants fonctionnels auront été spécifiés.

1.c.3.1. Les FICHIERS et les ENREGISTREMENTS

COMPOSANT = FICHIERS

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODEFI	Code d'identification du fichier					
LIBELFI	Libellé du fichier					
DESSIN-ENR	Définition du fichier					
TYPFI	Type du fichier					
ORGFI	Organisation du fichier					
SUPFI	Support du fichier					
VOLFI	Volume du fichier					
		ENREFI	ENREGISTREMENTS	Article associé au fichier		
		CLESFI	PROPRIETES	Liste des propriétés du fichier qui servent de critère de tri		
		UTIFI	OPERATEURS	Liste d'opérateurs qui utilisent le fichier en entrée ou en sortie		
		DOCASS	DOCUMENTS	Liste de documents qui engendrent le fichier, ou qui sont déduits du fichier		

REMARQUES :

(1) Un FICHIER est une collection d'articles de même nature. Il peut regrouper dans son enregistrement des propriétés relatives à un ou plusieurs ensembles qu'ils soient de base ou de relation.

Exemple :

Ensemble E_1	: Professeur	Ensemble E_2	: Matière
P_{11}	: Nom	P_{21}	: Titre
P_{12}	: Prénom	P_{22}	: Coefficient
P_{13}	: N° S.S.	P_{23}	: Niveau
P_{14}	: Adresse	P_{24}	: Contenu
P_{15}	: Sexe		
P_{16}	: Ancienneté		

Le Fichier F_1 regroupe les professeurs avec plus de deux ans d'enseignement dans l'établissement et ses matières enseignées.

Nom	Prénom	N° SS	Ancienneté	Titre	Titre ...

Enregistrement du fichier F_1

Les Fichiers de base seront traités par des programmes de type Cobol, Fortran, Assembleur, etc ...

(2) Le type d'un Fichier (TYPFI) peut être l'un des suivants :

1 - En entrée du traitement informatique
2 - En sortie du traitement informatique
3 - En entrée et en sortie du traitement informatique.

(3) L'organisation d'un Fichier (ORGFI) peut être du type :

1 - Séquentiel
2 - Séquentiel indexé
3 - Direct.

(4) Le support d'un Fichier (SUPFI) peut être :

1 - Bande
2 - Disque
3 - Cartes.

(5) Le volume d'un Fichier (VOLFI) est le nombre de caractères du Fichier.

(6) La définition du Fichier (DESSIN-ENR) permet la spécification des données (PROPRIETES) contenues dans son enregistrement, ainsi qu'une structuration de ces données avec un formalisme descriptif type Socrate ou PL/1.

COMPOSANT = ENREGISTREMENTS

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODENR	Code d'identification de l'enregistrement					
LIBELENR	Libellé de l'enregistrement					
DESSIN	Définition de l'enregistrement					
		ESPDONNEE	PROPRIETES	Liste de propriétés associées à l'enregistrement		
		FQUOI	FICHIERS	Liste des fichiers ayant cet enregistrement		
		DQUOI	DOCUMENTS	Liste des documents ayant cet enregistrement		

REMARQUES :

(1) Un ENREGISTREMENT est un regroupement des données (PROPRIETES). Il représente soit l'article d'un fichier, soit les données d'un document.

(2) Plusieurs fichiers et/ou documents peuvent avoir le même enregistrement.

1.c.3.2. La STRUCTURE DE BASE DE DONNEES

COMPOSANT = STRUCTURES DE BASE DE DONNEES

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODESTR	Code de la structure					
LIBELSTR	Libellé de la structure					
SOURCE	Définition de la structure					
DATE-1-STR	Date de la spécification de la structure					
DATE-2-STR	Date de confirmation de la structure par le responsable informatique					
DATE-3-STR	Date d'annulation de la structure					

REMARQUES :

(1) Les STRUCTURES DE BASE DE DONNEES de notre modèle de système d'information correspondent à l'implémentation physique de la structure logique de données. Il existe différents systèmes qui permettent la structuration de données sous forme de base de données et leur exploitation (cf. [30], [13], [11]).

(2) La définition (SOURCE) est le texte source de la structure de base de donnée.

(3) Parmi les différentes structures, chacune représente une traduction de la structure logique de données ; le responsable du projet devra choisir, à un moment donné de l'analyse, celle qui sera retenue par la suite (DATE-2-STR).

1.c.3.3. Les OPERATEURS

COMPOSANT = OPERATEURS (I)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	**Explication**	**Identificateur**	**Composant en relation**	**Explication**	Identificateur	Explication
CODEOP	Code d'identification de l'opérateur					
LIBELOP	Libellé de l'opérateur					
DEFOP	Définition de l'opérateur					
TYPOP	Type de l'opérateur					
USAGE	Usage de l'opérateur					
ETATOP	Etat de l'opérateur					
NATOP	Nature de l'opérateur					
ACCES-OP	Type d'accès à la base de données					
MODOP	Mode d'utilisation de l'opérateur					
LANGOP	Langage de programmation de l'opérateur					
SOURCE	Texte source de l'opérateur					

COMPOSANT = OPERATEURS (II)

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
SPECIF	Spécifications fonctionnelles de l'opérateur					
HISTOIRE	Description des éventuelles modifications de l'opérateur					
		REALISATEUR	PERSONNES	Analyste-programmeur responsable de l'implémentation de l'opérateur		
		ESPDONNEE	PROPRIETES	Liste de propriétés concernées par l'opérateur		
		COM-ASSOCIE	COMMANDES	Commande implémentée par l'opérateur		
		OP-APPELANTS	OPERATEURS	Liste de tous les opérateurs appelant cet opérateur		
		OP-APPELES	OPERATEURS	Liste de tous les opérateurs appelés par cet opérateur		
		FICH-CONCERNES	FICHIERS	Liste des fichiers concernés par l'opérateur	ACCES-FICH	Type d'accès au fichier

REMARQUES :

(1) Les OPERATEURS définissent, dans un formalisme sans ambiguïté, la logique des traitements demandés au niveau des commandes. Cette définition peut se faire sous deux formes :

a) En utilisant le formalisme des schémas de programmes qui ne correspond pas à un langage de programmation spécifique, mais qui permet la description structurée et sans ambiguïté de la commande.

b) En utilisant des langages de programmation qui explicitent d'eux-mêmes la logique du programme sans qu'il soit nécessaire de lui adjoindre de nombreux commentaires.

Les OPERATEURS doivent donc permettre, dans une phase d'analyse de la commande, la définition de sa logique et, dans une phase de programmation, l'implémentation exécutable du traitement correspondant.

(2) La caractéristique TYPOP définit le type d'Opérateur :

- saisie
- contrôle
- traitement
- édition.

Evidemment, un Opérateur peut être d'un type qui soit une combinaison quelconque des types mentionnés. Par exemple, saisie-contrôle ou bien saisie-contrôle-édition.

(3) Lorsque l'Opérateur est implémenté dans un langage de programmation, qu'il soit de type de gestion de base de données (langage SOCRATE) ou qu'il soit de type COBOL, FORTRAN, ASSEMBLEUR, etc ... Les caractéristiques suivantes permettent de le préciser :

* Langage (LANGOP) permet de définir le langage de programmation
* Nature (NATOP) permet de spécifier s'il s'agit d'un programme, d'une macro-instruction, d'une fonction, etc ...
* Type d'accès à la base de données (ACCES-OP) :

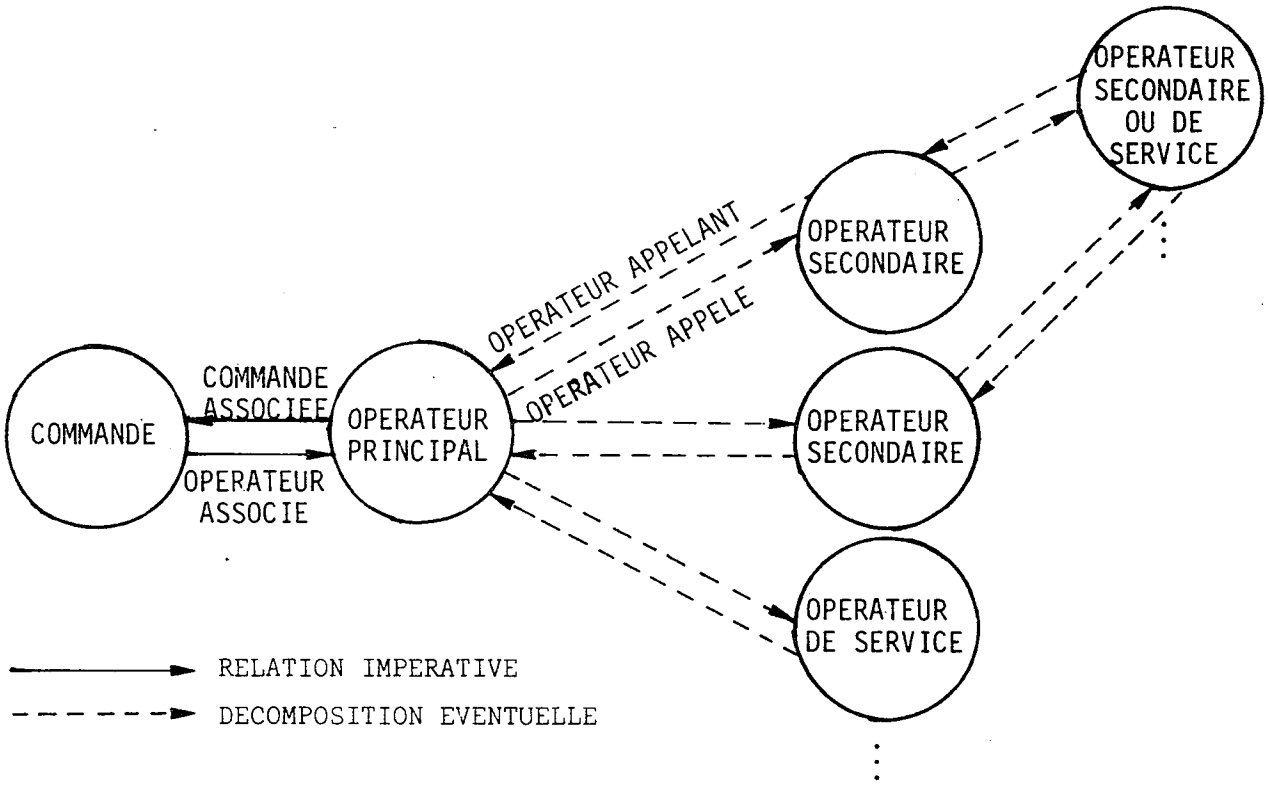
. Accès en lecture à la base
. Accès en écriture à la base
. Accès en lecture/écriture à la base
. N'accède pas à la base.

* Mode d'utilisation (MODOP) de l'opérateur :
  . Utilisation en mode conversationnel de l'opérateur
  . Utilisation en mode batch de l'opérateur
  . Utilisation en batch/conversationnel de l'opérateur.
* SOURCE pour répertorier le texte Source du programme
* Espace de données (ESPDONNEE) permet de répertorier les données utilisées ou produites par l'Opérateur
* Si l'Opérateur traite des données qui sont rangées dans un ou plusieurs fichiers, la liste de ces fichiers est enregistrée dans FICH-CONCERNES. L'accès à chacun de ces fichiers (ACCES-FICH) peut être :
  . Lecture
  . Ecriture
  . Lecture/Ecriture.

(4) Un Opérateur, s'il est très complexe, peut être l'objet d'une décomposition en Opérateurs plus élémentaires, ou bien il peut appeler plusieurs Opérateurs de la même façon qu'un programme peut demander l'exécution d'un ou plusieurs sous-programmes ou l'expansion de son texte source par appel à des macros-instructions.
La liste des Opérateurs plus élémentaires ou qui font l'objet d'un appel par l'opérateur est répertoriée dans OP-APPELES. Inversement, la liste de tous les opérateurs dont l'opérateur est un composant plus élémentaire ou appelé est désignée par OP-APPELANTS.

(5) Les informaticiens peuvent être amenés à se définir des OPERATEURS de SERVICE qui leur permettront une implémentation plus aisée et structurée des commandes. Ces opérateurs de service feront effectivement partie des opérateurs appelés par un ou plusieurs opérateurs. Evidemment, un opérateur de service peut appeler d'autres opérateurs de service.

(6) La figure suivante schématise la relation existante entre une commande et ses Opérateurs.



Toute commande est décrite formellement par un OPERATEUR PRINCIPAL qui peut être décomposé en plusieurs OPERATEURS SECONDAIRES et faire appel à plusieurs OPERATEURS DE SERVICE. Les Opérateurs secondaires ou de service peuvent être l'objet d'éventuelles décompositions.

La propriété USAGE permet de spécifier si l'Opérateur est principal, secondaire ou de service.

Pour un Opérateur principal, la commande associée est spécifiée par la propriété COM-ASSOCIE.

(7) A un moment donné, un opérateur doit se trouver dans un des états suivants (ETATOP) :

1) En cours d'analyse : l'analyste-programmeur fait l'étude de l'Opérateur à partir de sa définition.
2) Spécifié : l'analyse est terminée et la logique de l'Opérateur est définie par SPECIF.
3) Programmé : la programmation étant finie, on trouve le texte source du programme dans SOURCE.

4) Contrôlé-livrable : l'Opérateur étant programmé et tous ses fichiers étant définis, il faut pour qu'il soit dans cet état que tous les Opérateurs appelés soient aussi dans cet état. Autrement dit, tous les Opérateurs appelés à tous les niveaux doivent être programmés et ses fichiers définis.
5) Livré : l'Opérateur a été réceptionné par le responsable informatique.

L'état d'un opérateur principal conditionne les états (2,3,4,5,6) de la commande associée (cf. § 1.c.2.2.).

1.d. Le langage de description fonctionnelle

1.d.1. Principes généraux du langage de spécification LACSYST

Le langage de spécification LACSYST permet à l'analyse de définir une application informatique au niveau fonctionnel puis au niveau organique. Dans LACSYST, la spécification du projet informatique est une suite de description des différents composants qui constituent le projet. Ce langage est conçu pour décrire dans un ordre cohérent cette suite des composants. A chaque composant décrit précédemment (cf. § 1-c) est associé ce que l'on appelle une REGLE DE DESCRIPTION, qui définit l'ordre de spécification des différentes propriétés et relations des éléments d'un composant. A chaque règle de description correspond un graphique indiquant, sous une forme facilement compréhensible, cet ordre dans lequel les différentes valeurs des propriétés ou relations doivent être introduites.

Nous allons présenter maintenant l'ensemble des règles de description (notées ultérieurement R.D.) correspondant aux différents types de composants que nous avons déjà spécifié. Notons, dès maintenant, que chaque élément d'un composant, que ce soit une personne, un groupe, un module, une action, etc ... peut être déclaré, défini, modifié ou supprimé.

Déclarer : c'est donner le code et le libellé d'un élément.
Définir : c'est donner toutes les propriétés (sauf code et libellé) d'un élément déclaré :
- en valorisant les propriétés de cet élément,
- en déclarant de nouveaux éléments,

et/ou - en citant des éléments déjà déclarés.

Le langage LACSYST est composé de quatre types de R.D. :

1) Une R.D. de création pour créer un nouvel élément en donnant toutes les définitions de ces propriétés. On dira qu'un élément créé a été déclaré et défini.

Exemple : Une R.D. de création d'un Service Administratif (cf. § 1.c.2.1.) devra :

- Déclarer ce service :

Donner le code (CODSER) et le libellé (LIBELSER)

- Définir ce service :
  . Remplir : - la définition (DEFSER)
    - l'adresse (ADRESER)
    - et le téléphone (TELSER)
  . Attacher (citer) à ce service des personnes déjà déclarées (EMPLOYER)
  . Déclarer de nouvelles personnes et les attacher à ce service (EMPLOYER).

2) Une R.D. de mise à jour pour définir un élément déjà déclaré.

3) Une R.D. de modification d'un élément d'un composant, utilisée chaque fois qu'il est envisagé de modifier la description initiale des propriétés ou des relations de cet élément.

4) Une R.D. de suppression d'un élément lorsque celui-ci, initialement considéré comme utile dans le cadre du projet, est finalement rejeté après réflexion et doit donc être supprimé des spécifications.

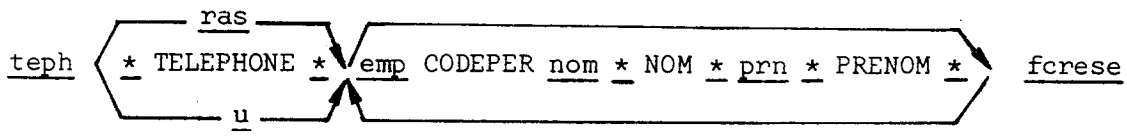
Correspondant aux différents types de composants définis au paragraphe 1.c., nous pouvons présenter maintenant les différentes règles de description qui leur sont associées dans le langage LACSYST.

1.d.2. R.D. de création des services administratifs et des personnes = crese, crepe

La R.D. de création des services administratifs permet la spécification des services concernés par l'application.

L'ordre d'introduction des données acceptées par cette R.D. est représenté par le graphe suivant :

crese CODESE * LIBELLE * def * DEFINITION * adr * ADRESSE *



Remarquons, une fois pour toute, que les propriétés de type "texte" d'un composant (DEFINITION, LIBELLE, ...) sont toujours encadrées par deux étoiles marquant le début et la fin du texte.

La spécification d'un service commence par le mot-clé crese suivi du libellé du service (LIBELSER).Le mot-clé def nous permet d'introduire la définition (DEFSER) du service administratif. Ensuite, l'adresse (ADRESER) du service est spécifiée en donnant le mot-clé adr puis le texte correspondant à l'adresse. Ensuite, le téléphone correspondant au service est défini par le mot réservé teph et trois situations différentes peuvent se présenter :

1) La spécification du téléphone par un texte entre deux étoiles (*)
2) Le service n'a pas de téléphone. Nous utiliserons alors le mot réservé "ras" (rien à signaler) qui est plus explicite qu'un blanc.
3) Le spécificateur désire reporter la spécification du téléphone. Nous indiquons par le mot réservé "u" (correspondant à ultérieurement) l'absence de spécifications.

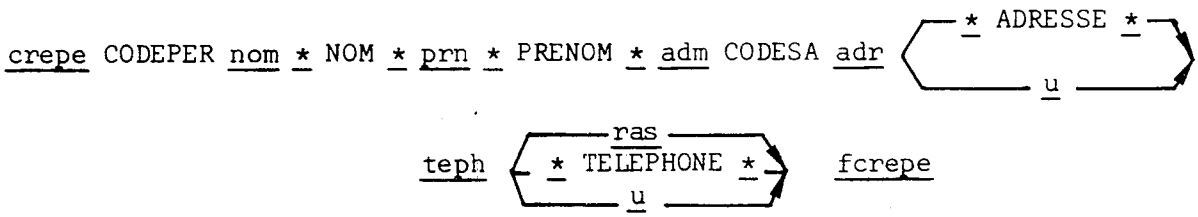
Ensuite, si le spécificateur connaît la liste des personnes appartenant au service et qui ont un rôle dans l'application, il est en mesure de la décrire en utilisant le mot-clé emp suivi du code de la personne (CODEPER), suivi du mot-clé nom et du nom de la personne (NOMPER), puis suivi du mot réservé prn pour donner le prénom de la personne (PRENOMPER). Tant qu'il y a des personnes à définir, l'utilisateur recommence avec le mot-clé emp. La fin de la description du service administratif est marqué par le mot réservé fcrese.

Exemple :

```
crese DIR01 * Direction de l'établissement scolaire *
      def * La direction de l'établissement et le service le plus haut dans
            la hiérarchie de l'établissement. Il est chargé de la direction
            et de l'administration de l'établissement ... *
      adr * 5, Rue Tristan Bernard - St Martin d'Hères *
      teph u
      emp P007 nom * Dupont * prn * Jean *
      emp P015 nom * Schael * prn * Coco *
fcrese
```

Pour pouvoir utiliser la R.D. de création des personnes (crepe) il faut que le service administratif auquel la personne est rattachée soit déjà décrit par crese (R.D. de création de services administratifs) et qu'elle ne soit pas encore citée dans d'autres règles de description.

Le schéma correspondant à la création d'une personne est le suivant :



La description de toute personne commence par le mot réservé crepe suivi du code de la personne (CODEPER). Ensuite, le mot-clé nom suivi du nom de la personne (NOMPER), le mot-clé prn pour spécifier le prénom (PRENOMPER). Ensuite, le mot-clé adm suivi du code d'un service administratif (CODESA) pour indiquer à quel service administratif la personne est rattachée. L'adresse est spécifiée par le mot-clé adr mais l'analyste a deux possibilités :

1) Spécifier cette adresse sous forme de texte
2) Indiquer par le mot réservé u qu'il la précisera ultérieurement.

Ces mêmes possibilités sont offertes à l'analyste quant au téléphone. Une troisième possibilité pour spécifier le téléphone :

3) La personne n'a pas de téléphone. On l'indique par le mot réservé ras.

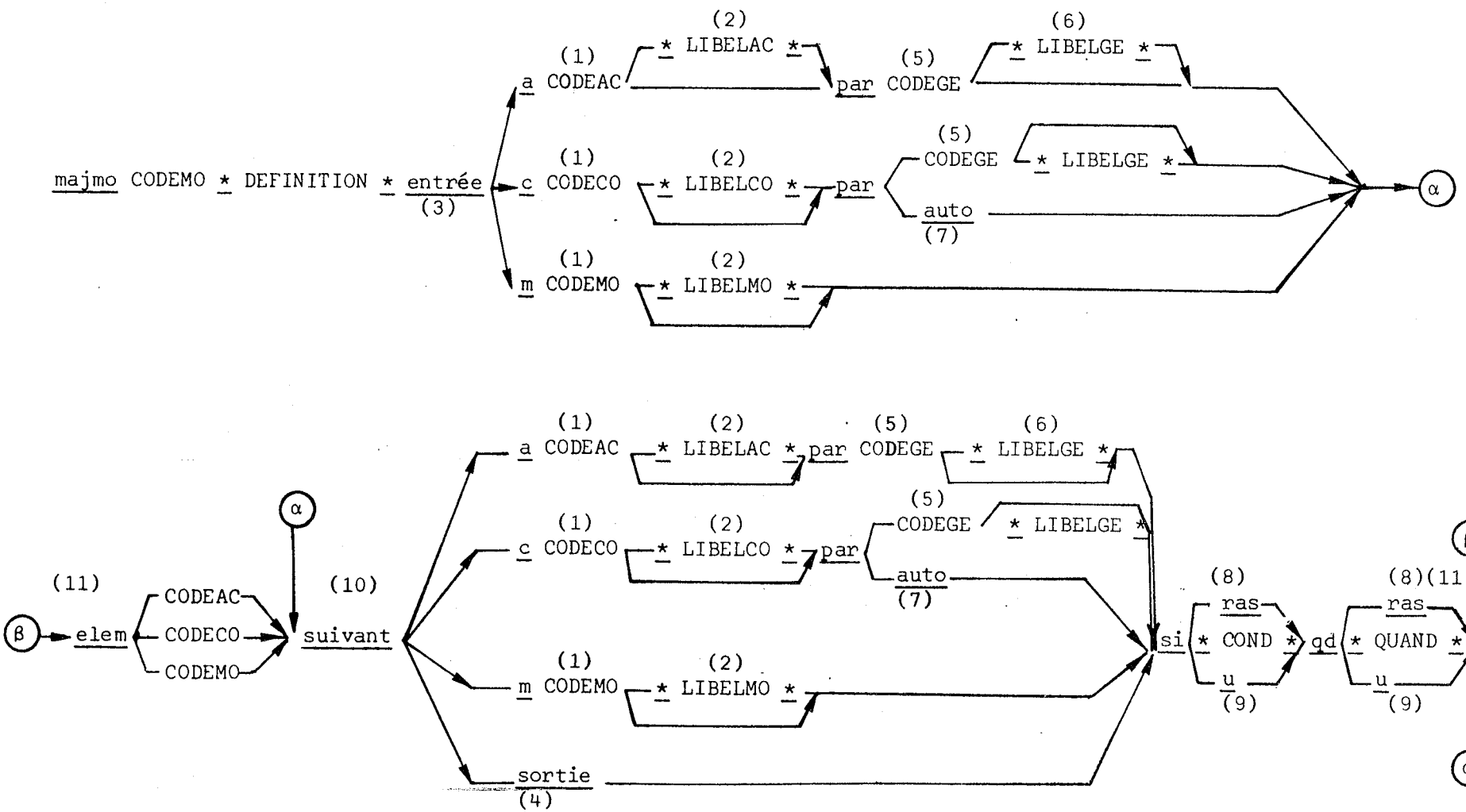
La description de la personne se termine par le mot-clé fcrepe.

Exemple :

```
crepe P009 nom * TARTAMPION * prn * LOUIS PHILIPPE *
           adm DIR01
           adr * 10, RUE BICONDE 38-GRENOBLE *
           teph u
fcrepe
```

1.d.3. R.D. de mise à jour des modules = majmo

Cette R.D. permet de définir la structure d'un module dont la déclaration a déjà été faite. Son code CODEMO et son libellé LIBMOD ont déjà été donnés. Elle permet de décrire la liste des éléments du module. L'ordre d'entrée des données acceptées par cette R.D. est représenté par le graphe suivant (la numérotation de ce cheminement permettra de commenter) :


majmo CODEMO * DEFINITION * entrée
(3)
a CODEAC
(1)
* LIBELAC *
(2)
par CODEGE
(5)
* LIBELGE *
(6)
c CODECO
(1)
* LIBELCO *
(2)
par
CODEGE
(5)
* LIBELGE *
auto
(7)
m CODEMO
(1)
* LIBELMO *
(2)
α
β
(11)
elem
CODEAC
CODECO
CODEMO
α
(10)
suivant
a CODEAC
(1)
* LIBELAC *
(2)
par CODEGE
(5)
* LIBELGE *
(6)
c CODECO
(1)
* LIBELCO *
(2)
par
CODEGE
(5)
* LIBELGE *
auto
(7)
m CODEMO
(1)
* LIBELMO *
(2)
sortie
(4)
si
ras
* COND *
u
(9)
(8)
qd
ras
* QUAND *
u
(9)
(8)(11)
β
fmajmo
(10)
α

Toute description d'un module commence par le mot-clé majmo, suivie du code du module CODEMO et de sa définition (DEFMO). Ensuite, c'est la description des composants du module.

Description des éléments d'un module :

* Les éléments d'un module peuvent être des actions (a), des commandes (c) et/ou des modules (m). Au moment de la description du module, certains de ces éléments ont été déjà déclarés.En ce cas, dans le processus de description des éléments, il suffira simplement de citer son code (1). Par contre, dans le cas où il s'agit de la déclaration d'un nouvel élément on donnera son code (1) et son libellé (2).

NOTE :

Un élément peut être en particulier déjà décrit comme élément-suite d'un autre élément dans le même module.

* Il est important pour pouvoir reconstituer automatiquement l'ordre d'enchaînement des éléments aussi bien que pour pouvoir éventuellement le modifier, d'indiquer lors de la définition des éléments quel est l'élément d'entrée du module (3), c'est-à-dire l'élément qui n'aura pas de PRECEDENTS et qui sera le point de départ de la structure du module. De même on indiquera le ou les points de sortie du module (4).

Quand l'élément correspond à une action ou à une commande, on indiquera par la clause "par" le code du groupe d'exécution (CODEGE) associé l'élément (5) suivi de son libellé LIBELGE (6) s'il s'agit d'un nouveau groupe d'exécution (déclaration).

Quand une commande sera exécutée automatiquement, c'est-à-dire que son activation ne dépend pas de l'initiative d'un groupe d'exécution, on l'indiquera par le mot réservé "auto" (7).

Définition de suites d'un élément :

Un élément peut avoir un ou plusieurs éléments SUITE, introduits par le mot réservé "suivant" auquel est associé le code de l'élément suivant ou le mot réservé "sortie" (4). Si l'élément suivant est nouveau, on donne immédiatement son libellé (2) et s'il s'agit d'une action ou d'une commande on utilisera la clause "par" décrite ci-dessus.

Les conditions de déclanchement d'un élément suivant un autre élément sont introduites par les mots réservés "si" et "qd". Le "si" permet de décrire sous forme de texte libre la ou les conditions de déclanchement autres que les conditions temporelles et le "qd" s'utilisera pour exprimer les conditions temporelles. Néanmoins, deux autres situations peuvent se présenter au moment d'exprimer une condition, qu'elle soit temporelle ou non :

1) Nous savons déjà qu'il n'y a aucune condition particulière à signaler (8). Nous utiliserons alors le mot réservé "ras".

2) Nous ne connaissons pas précisément ces conditions et nous reportons à une mise à jour ultérieure leurs spécifications (9). Nous indiquons par le mot réservé "u".

Tous les suivants d'un élément sont spécifiés les uns après les autres (10). Mais ils doivent ensuite être spécifiés en tant qu'éléments du module (11) sauf pour les cas des "sorties".

La fin de la description d'un module est identifiée par le mot clé "fmajmo".

Prenons l'exemple de la gestion des examens dans un établissement scolaire (cf. p. 1.39.) et décrivons la structure du module M10 avec "majmo".

```
majmo M10 * Ce module décrit fonctionnellement le processus trimestriel de
          la  pose des examens et de l'actualisation des notes *
     entrée m M11 * Elaboration du planning des examens *
              suivant a A12 * Affichage du planning *
                  par G1 * Service de Scolarité * si ras
                  qd * 15 jours avant le commencement des examens*
              suivant a A13 * Préparation des sujets d'examen *
                  par G2 * Groupe de professeurs * si ras qd ras
     elem A12 suivant m M14 * Déroulement des examens * si ras qd ras
     elem A13 suivant a A15 * Tirage des sujets d'examen *
                        par G3 * Service de tirage * si ras
                        qd * 4 jours avant le commencement des examens *
     elem A15 suivant m M14 si ras qd ras
     elem M14 suivant a A16 * Correction des examens * par G2
                        si ras qd ras
     elem A16 suivant a A17 * Perforation de résultats *
                        par G4 * Equipe de perforation *
                        si ras qd * 15 jours après le dernier examen *
     elem A17 suivant c C18 * Contrôles directs et indirects *
                        par G5 * Equipe d'exploitation * si ras qd ras
     elem C18 suivant m M19 * Correction d'erreurs *
                        si * il y a détection d'erreurs par la commande *
                        qd ras
              suivant c C20 * Mise à jour du fichier des notes *
                        par G5 si * pas d'erreurs * qd ras
     elem M19 suivant a A17 par G4 si ras qd ras
     elem C20 suivant c C21 * Edition des notes par groupe, par étudiant et
                                 par matière *
                        par G5 si ras qd ras
              suivant c C25 * Editions des tableaux récapitulatifs *
                        par G5 si ras qd ras
     elem C21 suivant   sortie  si ras qd ras
     elem C25 suivant   sortie  si ras qd ras
fmajmo
```

REMARQUES :

(1) Le lecteur a pu constater avec l'exemple de la gestion des examens exprimé soit sous forme de tableau (cf. p.1-39), soit par la R.D. "majmo", que des ambiguïtés peuvent se présenter dans la description de la structure modulaire au niveau des règles de composition.
Si nous représentons une partie du module M10 sous forme d'organigramme :

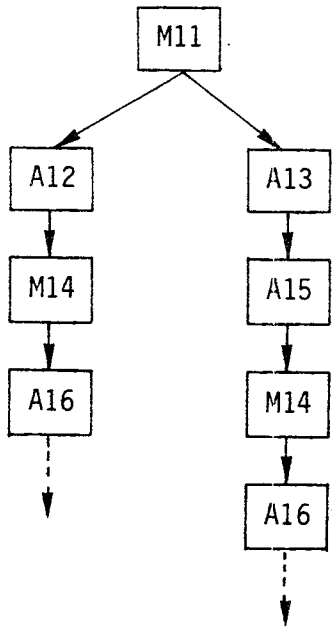


Figure 1-4

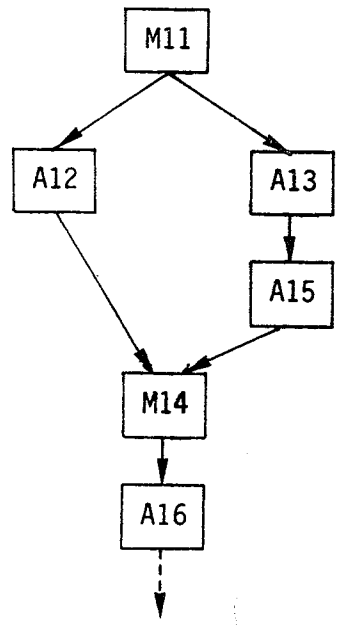


Figure 1-5

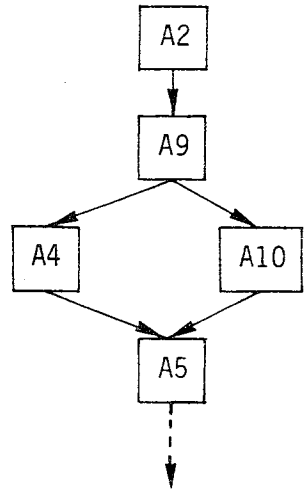
Est-ce la figure 1-4 ou la figure 1-5 qui représente la structure du module ? Pour supprimer toute ambiguïté, admettons les règles suivantes (voir exemple ci-dessous) :

1) Si dans la description successive d'éléments (A2,A9,A4,A10,A5) d'un module, plusieurs éléments (A4,A10) ont un suivant (A5) déclaré avec le même code, alors deux cas peuvent se présenter :

   a - Ce suivant (A5) n'a pas encore été déclaré comme élément du module. Ceci signifie que tous les éléments (A4,A10) doivent être exécutés pour que A5 soit exécuté.

Exemple :

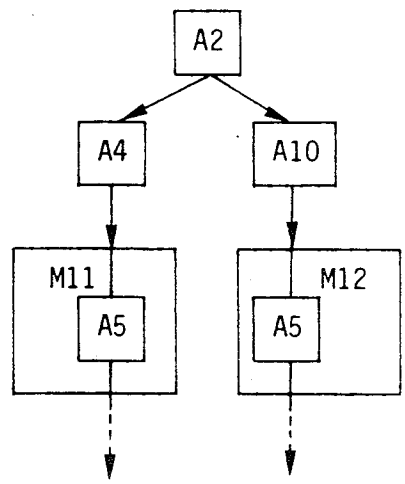
```
majmo M1
      entrée A2
            suivant A9
      elem A9
            suivant A4
            suivant A10
      elem A4
            suivant A5
      elem A10
            suivant A5
      elem A5
            suivant ...
      :
fmajmo
```



NOTE : Si l'on veut une double exécution indépendante de l'action A5 à partir de A4 et A10, on est obligé de déclarer des "modules artificiels". Ils sont "artificiels" car ils sont créés uniquement pour exprimer sans ambiguïté un mécanisme d'indépendance, ils ne correspondent fonctionnellement à rien de très précis dans le découpage du projet.

Exemple :

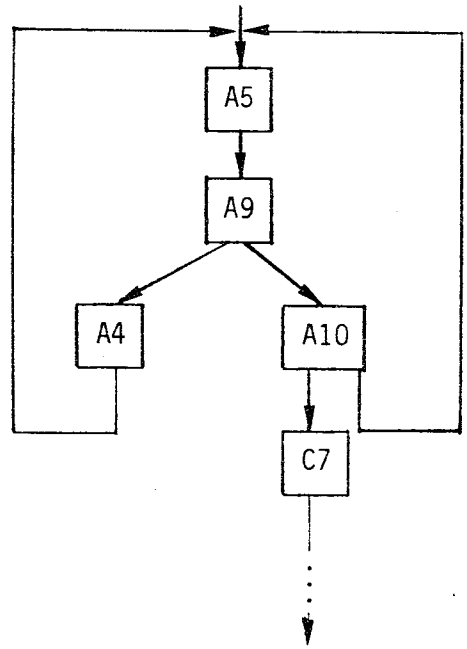
```
majmo M1
      entrée A2
            suivant A4
            suivant A10
      elem A4
            suivant M11
      elem A10
            suivant M12
      :
fmajmo
```



b - Les différents éléments pointent vers un suivant qui a déjà été déclaré comme un élément du module. C'est le cas de la boucle.

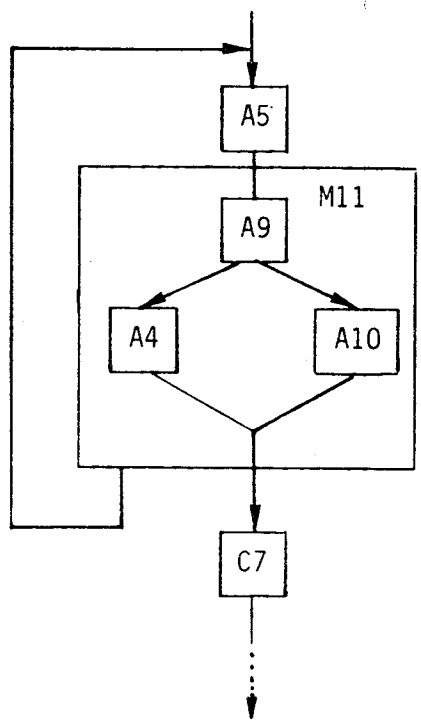
Exemple :

```
majmo M1
      entrée A5
            suivant A9
      elem A9
            suivant A4
            suivant A10
      elem A4
            suivant A5
      elem A10
            suivant A5
            suivant C7
      elem C7
      :     suivant ....
      :
fmajmo
```



NOTE : Si l'utilisateur désire que la boucle vers A5 se fasse après l'exécution de A4 et de A10, il est obligé de déclarer le module M11 suivant :

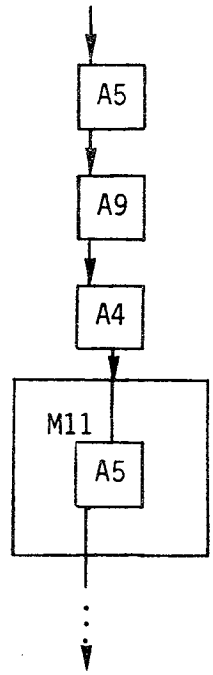
```
majmo M1
      entrée A5
            suivant M11
      elem M11
            suivant A5
            suivant C7
      elem C7
      :     suivant ...
      :
fmajmo
```



2) D'autres modules artificiels sont créés lorsque l'activation d'une même commande ou action est lancée à plusieurs endroits différents du module, et que l'on veut éviter une boucle.

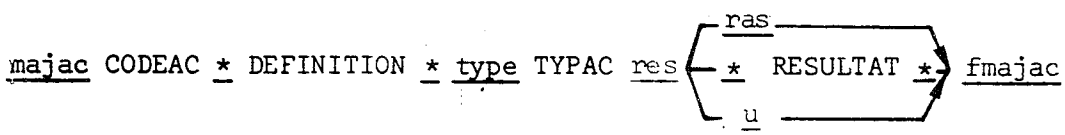
Exemple :

```
majmo M1
      entrée A5
            suivant A9
      elem A9
            suivant A4
      elem A4
            suivant M11
      elem M11
      :     suivant ...
      :
fmajmo
```



1.d.4. R.D. de mise à jour des actions, des commandes et des documents et des groupes : majac, majco, majdo, majgr

Avant de montrer les graphiques correspondant aux R.D. de mise à jour des actions, commandes et documents, il faut préciser que majac et majco ne peuvent être utilisés qu'à partir du moment où les objets qu'ils décrivent ont été déjà déclarés au niveau d'un majmo.
L'ordre d'entrée des propriétés d'une action est le suivant :



Le mot-clé majac indique qu'il s'agit de la R.D. de mise à jour des actions. Après le mot-clé majac, l'utilisateur donne la définition de l'action suivie du type de l'action qui doit être numérique et que nous avons appelé TYPAC lors de la description du composant action. Ensuite, le mot-clé res pour décrire le résultat attendu de l'action ; trois possibilités :

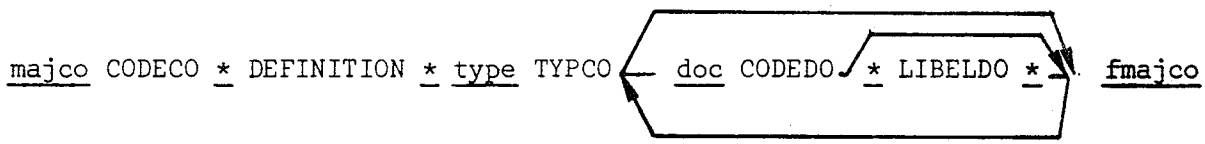
a) ras pour Rien A Signaler
b) u pour indéterminé et à préciser ultérieurement
c) La description du résultat en format libre.

La fin de la description est indiquée par le mot-clé fmajac.

Exemple :

majac A17 * L'équipe de perforation ayant en main les résultats des examens perfore les résultats de chaque élève dans chaque matière *
type 43
res * Un paquet de cartes contenant les résultats complets des examens (élève, matière, note) pour le trimestre correspondant *
fmajac

Le graphe correspondant à la règle de description de commandes est le suivant :



Cette règle commence par le mot-clé majco qui l'identifie, suivie par le code de la commande CODECO. L'utilisateur donnera ensuite la définition de la commande suivie immédiatement du type de la commande (TYPCO) :

1 = entrée-contrôle
2 = sortie.

Si l'analyste est en mesure d'associer un ou plusieurs DOCUMENTS à la commande, il donnera la liste de ces documents en utilisant le mot-clé doc suivie du code du document (CODEDO) et du libellé (LIBELDO). Il répétera cette opération sur tous les documents à associer. Le libellé du document (LIBELDO) pourra être omis si le document a été déclaré précédemment. La fin de majco est marquée par le mot réserve fmajco.

Exemple :

majco C18 * Cette commande est chargée de faire des contrôles directs sur les notes (numériques et entre 0 et 20) sur le code de matières (numérique), sur le trimestre en cours (numérique et entre 1 et 4), et de faire des contrôles indirects en vérifiant que l'élève existe bien dans le fichier des étudiants. Un document contenant les anomalies constatées sera édité par la commande *

type 1

doc D07 * Etat des anomalies *

doc D43 * Bordereau de perforation *

fmajco

La R.D. de mise à jour des documents majdo ne peut être utilisée que pour des documents qui ont été déclarés par la R.D. de mise à jour des commandes majco.

Le graphe associé est le suivant :

majdo CODEDO * DEFINITION * type TYPDO fmajdo

Toute description d'un document commence par le mot-clé majdo suivi du code du document (CODEDO). Ensuite la définition du document (DEFDO) sera donnée sous forme de texte libre. Immédiatement, on spécifie le type du document (TYPDO) représenté par :

1 = Entrée

2 = Sortie.

Finalement, le mot-clé fmajdo indique la fin de la règle de description.

Exemple :

majdo D07 * Ce document contient, sous forme de listing, les anomalies constatées par la commande C18. On trouvera la date, l'en-tête du document, le nom de l'étudiant, le type d'erreur et une reproduction de l'enregistrement correspondant. Il sera marqué sur le document "Pas d'erreur" lorsque la commande C18 ne détectera pas d'erreurs *

type 2

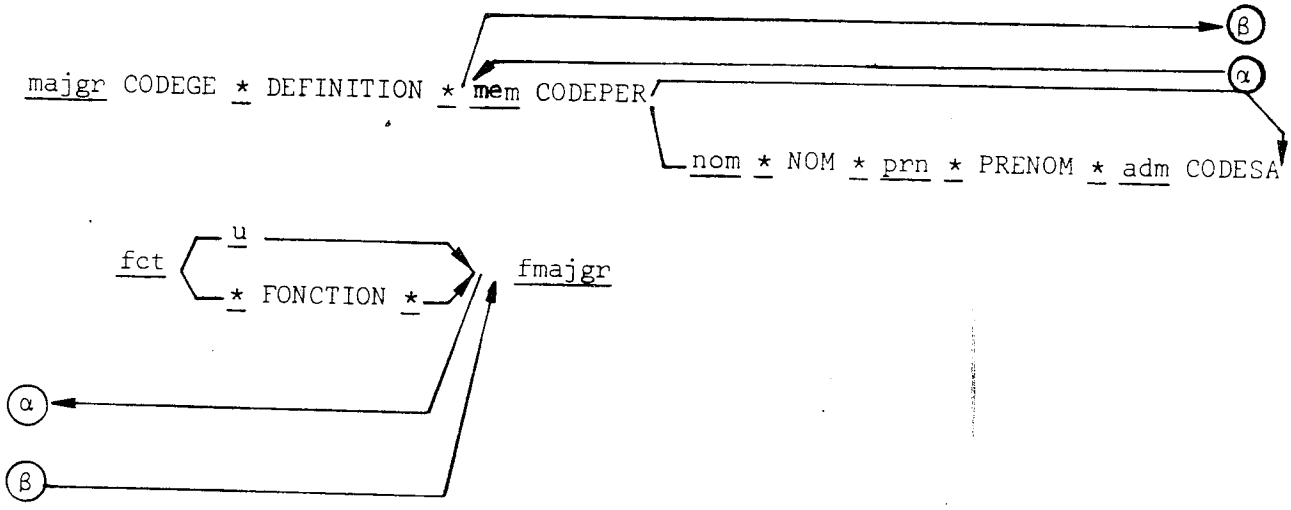
fmajdo

Les conditions d'utilisation de la R.D. de mise à jour des groupes d'exécution (majgr) sont :

1) que le groupe d'exécution à spécifier par majgr ait déjà été déclaré dans la description d'un module par majmo (R.D. de mise à jour des modules)

2) que les services administratifs des membres qu'on y associe aient déjà été décrits par crese (R.D. de création des services administratifs).

L'ordre d'introduction des données accepté par cette unité de description est représenté par le graphique suivant :



Le mot réservé majgr identifie la R.D. de mise à jour des groupes d'exécution. Il est suivi par le code du groupe (CODEGE) puis de la définition du groupe sous forme de texte libre. L'utilisateur a deux possibilités :

1) Il termine la description du groupe en utilisant le mot-clé fmajgr

2) Il donne la liste d'un ou plusieurs membres du groupe. Pour ce faire, il utilise le mot réservé mem suivi du code de la personne (CODEPER). Si la personne a déjà été créée par la R.D. de création de personnes (crepe) ou citée comme l'un des membres d'un service administratif par la R.D. de création des services administratifs (crese) alors l'utilisateur n'a pas besoin de rappeler son nom, prénom et administration. Par contre, si c'est la première citation de la personne, l'utilisateur après le code donnera le mot réservé nom suivi du nom de la personne (NOMPER), ensuite le mot-clé prn suivi du prénom (PRENOMPER) et finalement le mot-clé adm suivi du code du service administratif (CODESA) auquel la personne est rattachée. Pour décrire des autres membres, l'utilisateur recommencera le processus à partir du mot-clé mem. Pour terminer, il utilisera le mot-clé fmajgr.

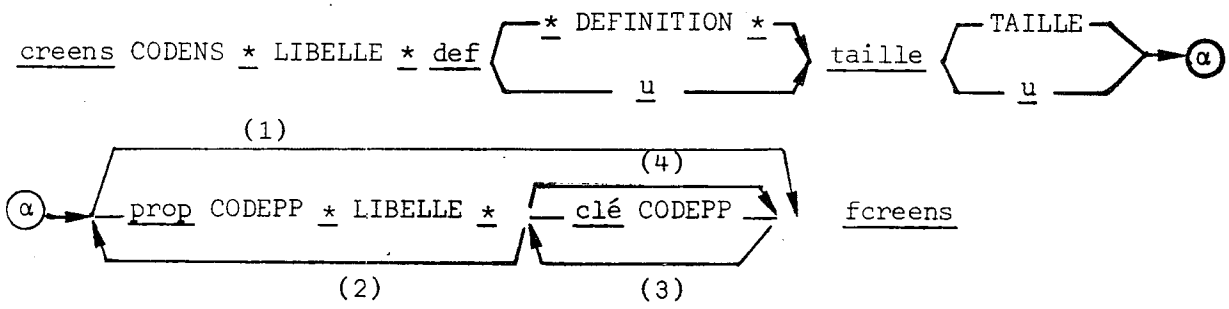
Exemple :

```
majgr G3 * Groupe d'exécution chargé de faire des photocopies, des tirages
           multiples, des reproductions, etc ... *
     mem P054 nom * LABICHE * prn * BRIGITTE * adm SVT04
     fct * Chargé du tirage *
fmajgr
```

1.d.5. R.D. de création des ensembles et des ensembles-relations = creens, crerl

R.D. de mise à jour des propriétés = majpp

Ces trois règles de description nous permettent la spécification des ensembles de base, des ensembles-relations et des propriétés, éléments qui constituent la structure logique de données de l'application.
Voyons d'abord l'ordre d'entrée des éléments dans la description d'un ensemble de base. Il est représenté par le graphe suivant :



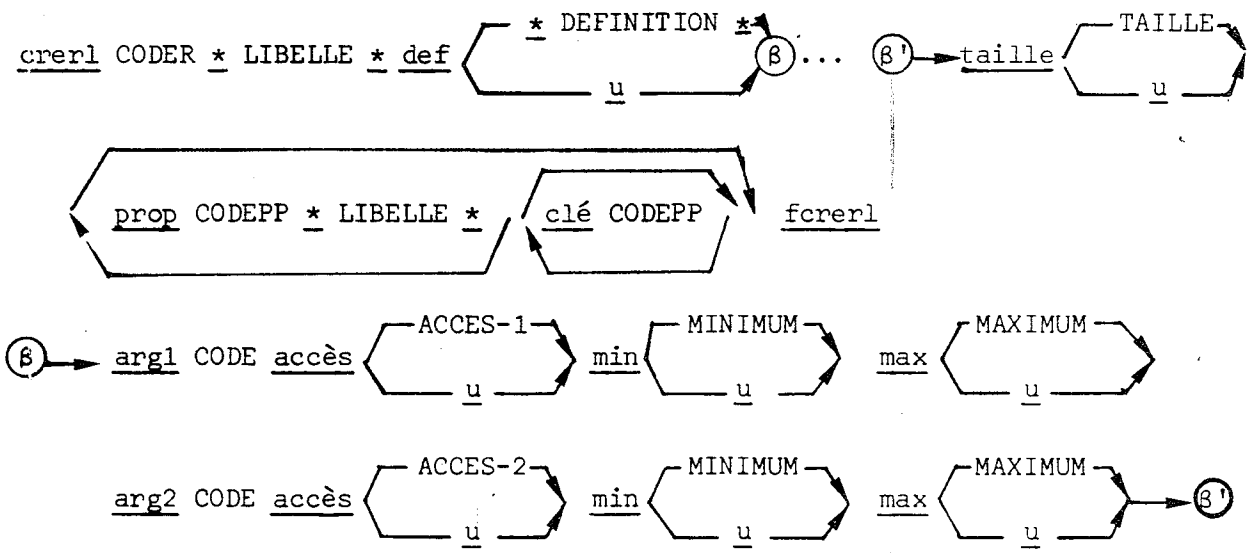
La description d'un ensemble commence toujours par le mot-clé creens. Ensuite, le code (CODENS) que l'on donne à l'ensemble suivi du libellé (LIBELENS). La spécification de la définition (DEFENS) commence par le mot-clé def suivi soit de la description sous forme de texte libre, soit par le mot-clé u pour indiquer une description ultérieure de la définition. Les mêmes options pour la description de la taille de l'ensemble (TAILLENS) qui doit commencer par le mot-clé taille suivi soit du cardinal exprimant la taille, soit du mot-clé u pour une spécification ultérieure. Si l'utilisateur est en mesure de lister les propriétés de ces éléments, il commencera par le mot-clé prop suivi du code qu'il veut assigner à la propriété (CODEPP) et du libellé (LIBELPP). Il réitèrera cette opération commençant par prop tant qu'il y aura des propriétés à déclarer (2). Ensuite, si l'une ou plusieurs de ces propriétés sont considérées comme des propriétés clés de l'ensemble, l'utilisateur pourra spécifier cette liste en donnant le mot réservé clé, puis le code de la propriété (CODEPP). Si aucune des propriétés est considérée comme clé, l'utilisateur après la spécification des propriétés termine la spécification de l'ensemble (4). De même qu'en passant par l'arc (1), l'utilisateur peut ne pas définir des propriétés, mais terminer la description de l'ensemble. Ceci se fait par le mot-clé fcreens.

Il faut remarquer que c'est seulement lorsque les propriétés d'un ensemble sont déclarées que l'on peut dire qu'elles en sont les clés.

Exemple :

```
creens E1 * Professeurs *
        def * Ensemble de personnes qui assurent un poste d'enseignant dans
              l'établissement scolaire. Ils doivent être titulaires du poste ou
              vacataires et assurer un minimum de 60 heures d'enseignement
              par année scolaire *
        taille 50
        prop p21 * Nom *
        prop p22 * Prénom *
        prop p23 * N° Sécurité Sociale *
        prop p27 * Catégorie *
        clé p23
fcreens
```

Le graphe associé à la R.D. de création des ensembles-relations est le suivant :



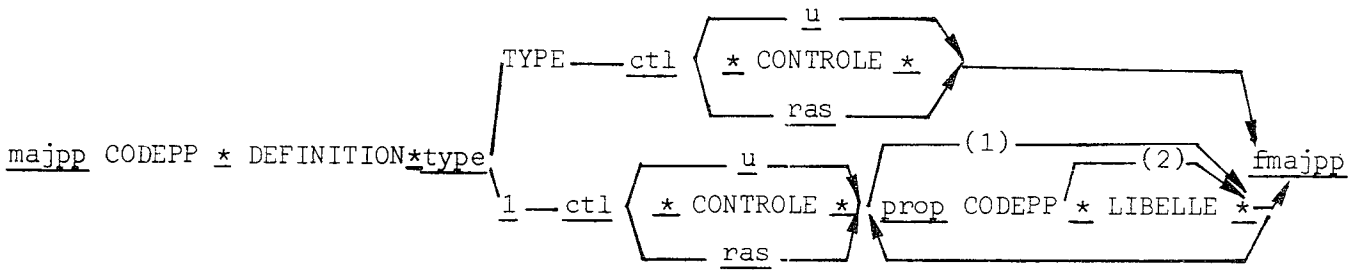
Le processus de spécification d'un ensemble-relation est le même que celui d'un ensemble de base, sauf qu'après la spécification de la définition (DEFER) et avant celle de la taille (TAILLER), l'utilisateur devra définir les arguments de la relation avec ses fonctions d'accès, ses maxima et ses minima ((β)...(β')).

Pour ce faire, on donne le mot-clé arg1 suivi du code de l'ensemble (de base ou relation = CODENS ou CODER) considéré comme premier argument de la relation. Ensuite, le mot-clé accès pour spécifier le nom que l'on donne à la fonction d'accès associée à ce premier argument (ACCES-1). La spécification de cette fonction d'accès pourra être faite ultérieurement ; ceci est indiqué par le mot-clé u. Le minimum associé à cette fonction d'accès est indiqué par le mot-clé min, puis le chiffre correspondant (MIN-1) ou bien la spécification ultérieure de ce chiffre par le mot-clé u. Même mécanisme pour définir le maximum de cette première fonction d'accès. Le processus pour définir le deuxième argument de la relation est similaire à celui employé pour la description du premier argument. Il faut remarquer que les arguments, arg-1 et arg-2, doivent être des ensembles déjà créés par l'une des règles de description creens ou crerl.

Exemple :

```
crerl R1 * Enseignement *
      def * R1 est l'ensemble défini par la relation existant entre les
            professeurs et les matières *
      arg1 E1 accès spécialité min 1 max 2
      arg2 E3 accès enseignant min 1 max 4
      taille 100
      prop P37 * Années de pratique *
      prop P39 * Niveau scientifique *
fcrerl
```

La description des propriétés se fait par la R.D. de mise à jour majpp. Son graphe est le suivant :



Cette R.D. de mise à jour ne peut être utilisée que si les propriétés dont elle permet la définition ont déjà été citées par les règles de description des ensembles (creens ou crerl) ou des propriétés (majpp).

Pour définir une propriété, on donne le mot-clé majpp qui permet d'identifier la règle de description. Ensuite,le code de la propriété (CODEPP) suivi de sa définition (DEFPP). Le mot-clé type permet deux options :

a) La propriété est élémentaire, alors on donne le type de valeur (TYPP) suivi du mot-clé ctl pour pouvoir spécifier le contrôle (CONTROLE). Cette spécification se fait soit ultérieurement (u), soit sous forme de texte libre, soit il n'y a rien à signaler (ras) comme contrôle.

b) La propriété est composée, alors on donne le mot-clé 1 suivi de la description du contrôle (cf. ci-dessus) et suivi de la citation éventuelle des propriétés composantes. En effet, on peut sauter la déclaration des propriétés compcsantes et le faire ultérieurement (1), ou bien on peut spécifier la liste en commençant par le mot réservé prop suivi du code de la propriété (CODEPP) et du libellé (LIBELPP) et ceci pour toutes les propriétés composantes que l'on veut déclarer (ce libellé pourra être omis (2) dans le cas d'une propriété déjà déclarée).

La spécification d'une propriété se termine par le mot-clé fmajpp.

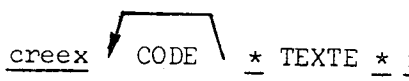
Exemple :

majpp p23 * Le numéro de sécurité sociale permet l'identification sans ambiguïté des professeurs, ainsi qu'un critère de tri *
type 1 ctl * Il faudra contrôler son existence, il doit être formé par 13 chiffres commençant toujours par 1 ou 2 *
prop P103 * Sexe *
prop P104 * Année de naissance *
prop P105 * Mois de naissance *
prop P106 * Département de naissance *
prop P107 * Commune de naissance *
prop P108 * Numéro d'ordre de naissance *
fmajpp

1.d.6. R.D. de création des explications = creex

Cette R.D. permet d'associer le texte d'une explication à un ou plusieurs composants. Pour pouvoir l'utiliser, il faut que tous les composants expliqués aient déjà été déclarés.

Le graphique d'entrée des données est le suivant :



Le mot réservé creex indique qu'il s'agit de la description d'une explication. L'utilisateur donnera la liste de tous les codes des composants auxquels il veut associer l'explication. Ensuite, il donne l'explication sous forme de texte encadré entre deux étoiles (*). Finalement, le mot-clé fcreex indique la fin de la règle de définition de l'explication.

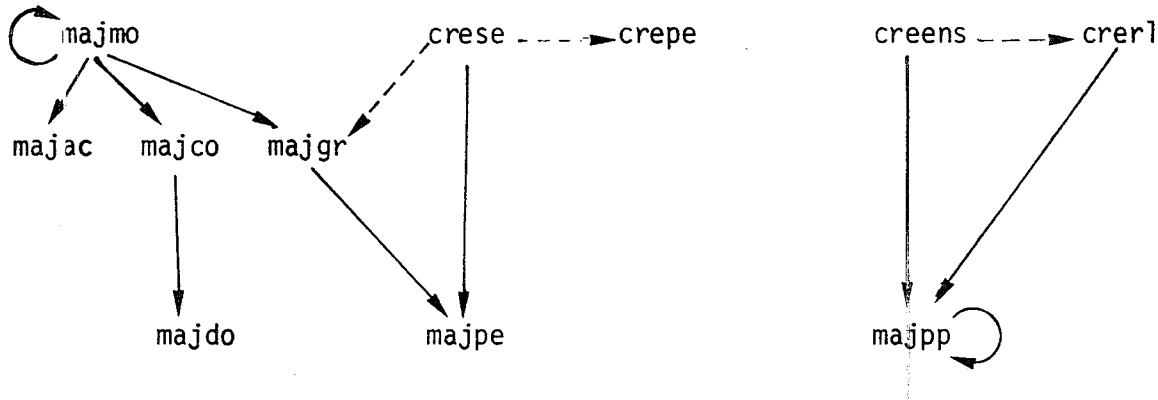
Exemple :

creex A13 A17 * Les professeurs ont officiellement 11 jours pour élaborer les examens et entre 14 et 21 jours pour les corriger *
fcreex

1.d.7. Remarques

(1) Le lecteur a pu constater que les différentes règles de description ne peuvent s'utiliser dans n'importe quel ordre. En effet, pour certains composants, on a associé des règles de création et pour d'autres des règles de mise à jour. Ceci est dû au fait que dans MACSI-1, on a considéré un ordre de spécification pour certains constituants : on ne peut pas, par exemple, spécifier les documents concernant une application sans avoir décrit au préalable les commandes qui traiteront ces documents. Le processus considéré comme cohérent pour spécifier une application par MACSI-1 sera décrit dans le chapitre 2. Néanmoins, on peut déjà décrire l'enchaînement des règles de description qu'on a vues dans les paragraphes précédents.

Voyons la figure suivante :



Commentaires :

a - Les flèches continues nous indiquent l'ordre dans lequel les règles peuvent être utilisées. L'utilisation des règles situées à l'origine de flèches entraînent l'initialisation d'éléments qui seront pris en compte par les règles pointées.

Par exemple, la figure nous montre que la mise à jour d'une action (majac) ne peut être effectuée que si cette action a été initialisée par un module (majmo). Ou bien on ne peut pas mettre à jour une personne si elle n'a pas été déclarée
- soit par la création d'un service administratif
- soit par la mise à jour d'un groupe d'exécution.

C'est dans une optique de conception descendante du projet que ces contraintes ont été adoptées.

b - D'autres options plus implicites sont aussi imposées, mais qui ont plutôt un caractère organisationnel. Elles sont signalées par les flèches pointillées. Par exemple, on dit que la création d'une personne ne peut s'effectuer que si le service administratif auquel elle est rattachée est déjà créé. De même, on dit que pour établir une relation entre deux ensembles (crerl), il est nécessaire que ces deux ensembles aient déjà été créés.

(2) Résumons dans le tableau ci-dessous les noms des différentes R.D. correspondant à une opération de création, de mise à jour, de modification ou de suppression d'un élément d'un composant du modèle fonctionnel.

Opération / Composant	Création	Mise à jour	Modification	Suppression
Service	crese		modse	supse
Personne	crepe	majpe	modpe	
Groupe		majgr	modgr	
Module		majmo	modmo	
Commande		majco	modco	
Action		majac	modac	
Ensemble	creens		modens	supens
Relation	crerl		modrl	suprl
Propriété		majpp	modpp	

Remarquons que toutes les cases de ce tableau ne sont pas remplies. Généralement, un composant qui peut être créé n'aura pas de R.D. de type mise à jour associé, mais il y aura une R.D. de type suppression. Inversement, à un composant C1 qui peut être déclaré lors d'une opération sur un élément d'un autre composant C2, on n'associe pas une R.D. de suppression sur C1. Cette suppression d'un élément de C1 sera réalisée par une modification ou une suppression d'un élément de C2.

Une R.D. de type modification est associée à tous les composants.

(3) Pour spécifier un ou plusieurs constituants, l'analyste se sert des différents graphiques qui constituent le langage LACSYST. Il utilise des documents (document au sens MACSI-1 = bordereaux de perforation pour une entrée batch, clavier du télétype pour une entrée conversationnelle) pour donner ses spécifications. On a pris l'option de ne pas imposer à l'analyste des documents zonés où on définissait des colonnes et des marges de commencement et de fin, mais plutôt des documents à format libre (peu importe si l'on rentre les spécifications les unes à la suite des autres sur une ou deux lignes ou si par contre on utilise une ligne par caractéristique) qui laissent l'analyste beaucoup plus libre dans sa description et qui n'est pas source d'éventuelles erreurs dues à un mauvais positionnement des données. Par contre, nous avons été obligé d'introduire de nombreux mots-clé pour démarquer ces spécifications (def, type, ..., *).

CHAPITRE 2

SPECIFICATIONS FONCTIONNELLES DE MACSI-1 PAR MACSI-1

Après avoir décrit les différents constituants d'un système d'information dont il est demandé la spécification au niveau fonctionnel, après avoir indiqué selon quelles règles de description ces spécifications doivent être rédigées, le lecteur aura peut-être une certaine impression de confusion. En effet, il n'a pas été indiqué clairement jusqu'ici dans quel ordre utiliser ces règles. Surtout, rien n'a été précisé concernant les exploitations qui seront faites par ordinateur des descriptions fournies par les analystes.

Il faut donc préciser maintenant l'organisation générale qui doit être adoptée dans la mise en oeuvre de MACSI-1.

Pour ce faire, nous pourrions prendre une présentation classique d'exposé. Mais nous pouvons aussi envisager d'utiliser les règles de description présentées au chapitre 1 pour présenter l'organisation de MACSI-1 lui-même.

C'est cette solution que nous avons retenue pour ce chapitre. Ce choix nous a été dicté pour :

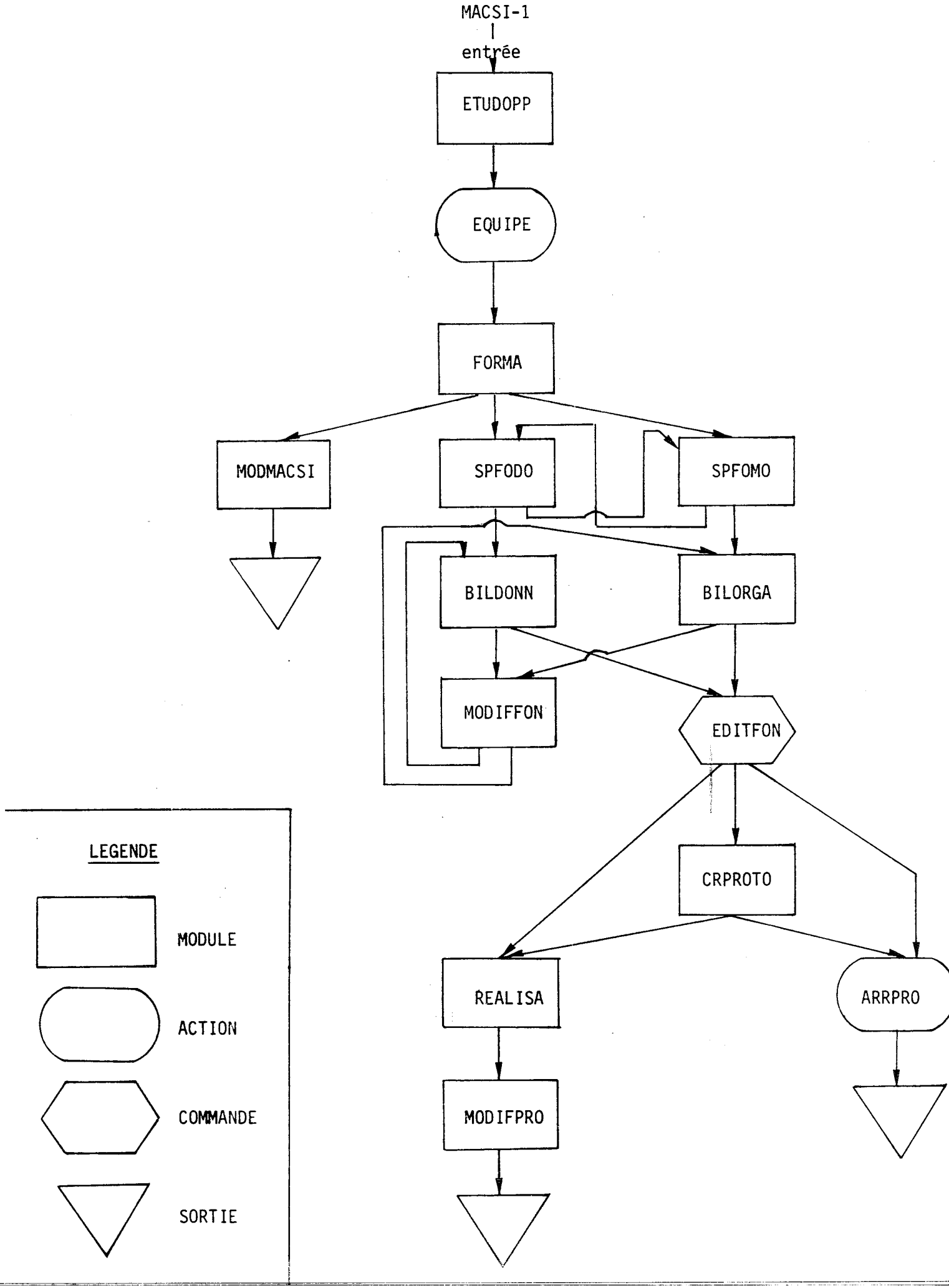
- Proposer à notre lecteur, grâce à un cas concret d'une complexité plus grande que les petits exemples présentés au chapitre 1, une illustration des règles de description du langage LACSYST.

- Ce faisant, permettre une évaluation des possibilités et des limites de ce langage.

- Mettre en relief certains aspects méthodologiques d'utilisation de ce langage qui ne peuvent pas être présentés de façon abstraite. C'est pourquoi dans ce chapitre, certains paragraphes, cadrés à gauche par un double trait plein, mettent en évidence, au moment où ils se posent, ces aspects de méthode.

Toutefois, cette description fonctionnelle de MACSI-1 n'est pas complète. Comme il était souhaitable de ne pas répéter la présentation de certains concepts exposés déjà au chapitre 1 et comme, d'autre part, le chapitre 3 a pour objet d'expliquer de façon détaillée la manière d'utiliser MACSI-1 dans la phase d'analyse organique et de programmation d'une application informatique, nous ne présentons ici qu'une partie des spécifications fonctionnelles de MACSI-1 par MACSI-1.

SCHEMA GENERAL DE LA STRUCTURE DE MACSI-1



majmo MACSI-1

* MACSI-1 est une méthode d'analyse et de documentation d'un projet informatique assistée par ordinateur qui retient comme objectifs essentiels :

1°) description fonctionnelle de l'organisation administrative par des gestionnaires suivant un processus d'analyse descendante.

2°) description de la logique des données indépendamment de tout choix ultérieur sur les méthodes de stockage de ces données en mémoire et des procédures d'accès associées.

3°) Standardisation des spécifications organiques pendant la réalisation informatique.

4°) Stockage dans une base de données de ces descriptions pour pouvoir par programmes assurer :

- la documentation automatique des spécifications du projet,
- des diagnostics automatiques d'incohérence dans la conception du projet.

5°) Réalisation de la base de données et des programmes de MACSI-1 permettant facilement leur modification si des modifications de la méthodologie associée sont souhaitées.

*

entrée m ETUDOPP * étude d'opportunité du projet *
suivant a EQUIPE * constitution de l'équipe responsable du projet *
par GO * administrateurs de la base MACSI-1 * si ras qd ras
elem EQUIPE suivant m FORMA
* formation de l'équipe au maniement de MACSI-1 *
si ras qd ras
elem FORMA suivant m MODMACSI * modification de MACSI-1 *
si * la méthode demande quelques aménagements * qd ras
suivant m SPFODO * spécifications fonctionnelles des données *
si * la description logique des données dans MACSI-1 est jugée adéquate * qd ras
suivant m SPFOMO * spécifications fonctionnelles des modules *
si * l'analyse modulaire de l'organisation administrative est retenue * qd ras

elem SPFODO suivant m BILDONN * Bilan sur la logique des données *
si * à l'initiative du chef de projet * qd ras
suivant m SPFOMO si * les analystes veulent passer de la description des modules à celle des données *

elem SPFOMO suivant m BILORGA * Bilan sur la description modulaire de l'organisation administrative * si * à l'initiative du chef de projet * qd ras
suivant m SPFODO * si * les analystes souhaitent passer de la description fonctionnelle des données à celle de l'organisation administrative * qd ras

elem BILDONN suivant m MODIFFON * modification des spécifications fonctionnelles *
si * le bilan sur les données diagnostique des anomalies * qd ras
suivant c EDITFON * édition définitive du dossier fonctionnel *
par G2 * le chef de projet *
si * le chef de projet donne son accord aux spécifications fonctionnelles du projet * qd ras

elem BILORGA suivant m MODIFFON
si * des incohérences détectées sur la description des modules, actions et commandes * qd ras
suivant c EDITFON par G2 si u qd ras

elem MODIFFON suivant m BILDONN si * dans tous les cas * qd ras
suivant m BILORGA si * dans tous les cas * qd ras

elem EDITFON suivant m CRPROTO * création d'un prototype du projet *
si * le chef de projet juge souhaitable et possible de construire un prototype du projet préalablement à la version opérationnelle * qd ras
suivant m REALISA * réalisation de la première version opérationnelle du projet *
si * un prototype du projet n'est pas souhaitable ou n'est pas réalisable dans des conditions de coût et de délai acceptables * qd ras
suivant a ARRPRO * arrêt du projet * par G1 * direction du projet * si * la définition fonctionnelle du projet démontre l'impossibilité de le réaliser dans de bonnes conditions * qd ras

elem CRPROTO *suivant* a ARRPRO *par* G1

si * *les résultats du prototype n'ont pas prouvé l'intérêt ou la faisabilité du projet* * *qd ras*

suivant m REALISA *si* * *les résultats du prototype permettant à la direction d'envisager la réalisation de la première version opérationnelle de l'application informatique* * *qd ras*

elem ARRPRO *suivant sortie si ras qd ras*

elem REALISA *suivant* m MODIFPRO * *modification du projet* *

si u qd * *dès que des utilisateurs du projet demandent des modifications de celui-ci* *

elem MODIFPRO *suivant sortie si ras qd ras*

fmajmo

Il y a volontairement une erreur dans cette description du module MACSI-1 : l'élément MODMACSI n'a pas de "suivant". Cette erreur dans l'application de la règle de description d'un module peut être facilement détectée par programme. C'est là un exemple caractéristique de diagnostic automatique de cohérence. Il sera repris dans le module BILORGA.

majgr G1 * *la direction du projet est en général composée du chef de projet mais surtout de représentants des services qui financent sa réalisation et de représentants des principaux services qui seront les futurs utilisateurs du projet quand il fonctionnera* *

fmajgr

majac ARRPRO

* *L'arrêt du projet peut être décidé si l'analyse fonctionnelle conduit à conclure à une faisabilité impossible de la réalisation technique dès la fin de l'analyse fonctionnelle, ou bien à un bilan coût-efficacité de l'application médiocre. La réalisation d'un prototype du projet en particulier peut permettre de conclure que les moyens informatiques proposés aux services utilisateurs ne leur apportant que des possibilités limitées dans leur travail* *

type 13

res * l'arrêt du projet doit être accompagné d'un rapport de synthèse dans lequel figure :

- le budget de développement du projet s'il avait été réalisé complètement et le planning associé
- une évaluation des dépenses déjà réalisées pour l'étude
- une présentation des motifs pris en compte pour arrêter le projet *

fmajac

majac EQUIPE

* La formation de l'équipe de réalisation du projet doit intervenir dès le démarrage de l'étude pour préciser quelles sont les différentes personnes qui participent à la conception.

Les principales fonctions à distinguer sont les suivantes :

- la direction du projet (G1) qui le finance et décide éventuellement son arrêt
- le chef de projet (G2) qui assure la coordination du projet, contrôle son avancement et l'état de sa documentation. Il rend compte à la direction du projet
- les personnes détachées des principaux services concernés par le projet pour participer à sa conception au niveau fonctionnel (G3). On les appelle quelquefois "correspondants informatiques" du projet
- le secrétariat technique du projet (G4)
- le responsable informatique qui assure la coordination et le suivi des tâches de réalisation technique (G5)
- les analystes-programmeurs qui implémenteront le projet (G6). Ils rendent compte au responsable informatique

*

type 13

res * Note spécifiant l'organisation de l'équipe *

fmajac

majco EDITFON

** Edition du dossier d'analyse fonctionnelle qui comprendra :*
- tous les modules avec leurs éléments,
- toutes les actions, les commandes et leurs documents,
- les services,
- les groupes fonctionnels,
- les ensembles, leur relation et leurs propriétés.
L'ensemble de ces listes sera classé par ordre alphabétique. On éditera à la fin un glossaire de tous les codes.
*L'édition se fera en autant d'exemplaires que de membres de l'équipe de conception **
type 1
*doc DOSSANFO * dossier d'analyse fonctionnelle **

fmajco

Depuis la description initiale de MACSI-1 jusqu'ici, nous nous sommes situés dans le cadre de SPFOMO appliqué à MACSI-1. Nous souhaitons maintenant apporter quelques premières précisions sur les informations manipulées dans MACSI-1. Nous passons à l'exécution de SPFODO dans le cadre du projet MACSI-1.

*creens COMPOSANT * Différents types de composants d'un projet **
*def * En distinguant dans un système d'information différents types de composants (des personnes, des actions, des modules, des documents, etc ...), il est évident qu'il faudra, pour chaque élément de n'importe quel type, pouvoir l'identifier.*
Cette identification dans tous les cas portera sur :
- un code discriminant
- un libellé très court qui donnera la signification très grossière de l'élément
*- une définition sous forme d'un texte aussi long que nécessaire pour préciser totalement la signification de l'élément **
taille 3000
*prop CODE * code de l'élément **
*prop LIBELLE * libellé de l'élément **
*prop DEFINITION * définition de l'élément **
clé CODE

fcreens

creens SERADM * Services administratifs, commerciaux et de production *
def * Tous les services (même s'ils portent d'autres noms comme division, unité, département, etc ...) qui seront concernés, d'une manière ou d'une autre, par le bon fonctionnement de l'application informatique lorsque celle-ci sera opérationnelle. Ces services sont en général décrits dans l'organigramme hiérarchique de l'entreprise ou de l'administration. Chaque personne concernée par le projet a un rattachement unique à un de ces services *
taille 20
fcreens

creens PERSONNE * personnes participant au fonctionnement du projet *
def * Ces personnes sont toutes celles qui, à un titre quelconque dans leur service, sont concernées par le fonctionnement du projet quand celui-ci sera en exploitation. Elles devront être formées pour les tâches qui leur seront confiées. Ces tâches sont celles des groupes d'exécution auxquelles elles seront rattachées *
taille 90
prop NOM * nom de la personne *
prop PRENOM * prénom de la personne *
prop TELEPHONE * numéro de téléphone *
fcreens

majpp TELEPHONE * Il s'agit du numéro de téléphone où l'on peut joindre le plus facilement une personne pendant les heures de bureau. Il peut être différent de celui de son service si cette personne est détachée dans un autre service *
type mot
ctl ras
fmajpp

creens GROUPE * Groupe d'exécution *
def * Les groupes d'exécution sont introduits dans le modèle pour distinguer un ensemble de personnes chargées d'exécuter une liste de tâches dans le cadre du projet. En général, ce groupe est soit une partie seulement d'un service, soit un ensemble de personnes qui sont rattachées à différents services.
La notion de groupe d'exécution ne peut donc pas se confondre avec celle de services administratifs *
taille 200
fcreens

creex PERSONNE SERVICE GROUPE
* Dans la progression de l'analyse d'un projet, on spécifiera dans l'ordre :
- d'abord les services administratifs concernés,
- ensuite, au fur et à mesure que se précise l'organisation du projet, sont définis les groupes d'exécution,
- enfin, pour le lancement du projet, on indiquera toutes les personnes concernées et leur service de rattachement. Les obligations de chaque personne sont celles des groupes d'exécution auxquels elle sera associée *
fcreex

crerl COMPSER * Les services administratifs sont un type de composant *
def * L'ensemble des services administratifs sont un type de composant et ont donc, à ce titre, un code, un libellé et une définition *
arg1 COMPOSANT accès u min 0 max 1
arg2 SERADM accès u min 1 max 1
taille 20
fcrerl

Cette relation COMPSER n'a d'autre objectif que de permettre la définition d'un CODE, d'un LIBELLE et d'une DEFINITION pour les services SERADM. En effet, ces trois propriétés étant définies au niveau de l'ensemble des COMPOSANTS, elles leur sont spécifiques et ne peuvent donc pas être des propriétés des autres ensembles. Cet exemple illustre les avantages et inconvénients de la règle fixée à la remarque (5) de la page 1.56 concernant l'association d'une propriété avec un ensemble et un seul.
On aurait pu procéder comme au chapitre 1 en ne déclarant aucun ensemble COMPOSANT et en associant par exemple :
à l'ensemble SERADM les propriétés CODESER LIBELSER et DEFSER
à l'ensemble GROUPE les propriétés CODEGE LIBELGE et DEFGE

Dans ce cas, rien n'indique explicitement que CODESER et CODEGE sont deux formes d'un code qui doit être défini pour tout composant et cette ellipse peut être dangereuse. Par contre, il ne faut pas décrire, comme ci-dessus, toutes les relations permettant d'indiquer que les SERADM, les groupes, etc ... sont des composants et, à ce titre, ont un code, un libellé et une définition.

Notons enfin qu'il serait fastidieux de continuer à définir toutes les informations manipulées dans MACSI-1. Avec l'exemple précédent, le lecteur aura certainement remarqué que la présentation au chapitre 1 sous forme de tableaux de toutes les propriétés des composants et de leurs relations a été choisie pour faciliter la traduction du contenu de ces tableaux avec les trois commandes CREENS, CRERL et MAJPP.

Reprenons maintenant la description de l'organisation de MACSI-1 en décomposant le module SPFOMO qui est, pour l'instant, simplement déclaré dans le module MACSI-1.

majmo SPFOMO

* *Spécifications fonctionnelles de l'organisation du projet. Elle permet de décrire l'enchaînement dynamique des actions non automatisées et des commandes de traitement automatique de l'information selon un processus d'analyse descendante.*
Dans la description, chaque module, déclaré comme élément d'un module qui le contient, doit être à son tour détaillé jusqu'à n'avoir au niveau le plus élémentaire plus que :
- *des commandes de traitement automatique de données,*
- *des actions qui, dynamiquement, doivent précéder ou suivre les commandes,*
- *les groupes chargés de l'exécution de ces commandes et ces actions.*

Parallèlement peut être décrite la structure administrative des services qui collaboreront au projet lorsqu'il sera en exploitation.
*Enfin, pour chaque commande définie dans la description modulaire, il faut préciser les caractéristiques du document qui lui est associé. La description de l'application commence par la description du module associé à l'application considérée elle-même comme un module qui contient tous les autres **

*entrée m MAMO * mise à jour du module associé à l'application **
*suivant a APPRECISER * Choix des composants déclarés à préciser * par G3*
** groupe des correspondants chargés de l'analyse fonctionnelle * si u qd ras*

elem APRECISER
*suivant c CREEX * explication commune à plusieurs composants * par G3 si ras qd ras*
*suivant c MAJMO * description d'un module * par G3*
*si * pour tout module déclaré comme élément d'un module et qui n'est pas encore précisé **
*qd * dès que la structure de ce module peut être précisée **
*suivant c MAJCO * description d'une commande * par G3*
*si * une commande déclarée comme élément d'un module n'est pas encore elle-même décrite * qd ras*

*suivant c MAJAC * description d'une action * par G3*
*si * une action déclarée dans un module comme un de ses éléments peut être précisée * qd ras*

*suivant c CRESE * description d'un service * par G2*
*si * pour tout service concerné par le projet * qd ras*

*suivant c MAJGR * description de groupe d'exécution * par G3*
*si * pour tout groupe associé dans un module à une action ou une commande * qd ras*

*suivant sortie si * les analystes veulent plutôt définir des données de l'application dans le cadre de SPFODO * si ras qd ras*

*elem MAJMO suivant c BILPROV * bilan provisoire des composants spécifiés incomplètement * par G2*
*si * dès que les participants au projet dans G3 en expriment le besoin car ils ne peuvent plus savoir de mémoire quels sont les éléments de modules déclarés qui ont été décrits ultérieurement et ceux qui restent à préciser **
*qd * au moins tous les 15 jours pour faire le point sur l'avancement du projet **
*suivant a APRECISER par G3 si * en tant que de besoin * qd ras*

elem MAJCO suivant a APRECISER par G3 si ras qd ras
suivant c MAJDO par G3
*si * pour les documents déclarés dans la description d'une commande * qd u*

elem MAJAC suivant a APRECISER par G3 si ras qd ras

elem CRESE suivant a APRECISER par G3 si ras qd ras

elem MAJDO suivant a APRECISER par G3 si ras qd ras

elem MAJGR suivant a APRECISER par G3 si ras qd ras

*elem BILPROV suivant c CREPE * identification d'une personne * par G2*
*si * utilisée uniquement par le chef de projet, cette commande permet d'identifier les personnes dont on sait, dès le début de l'étude fonctionnelle du projet, qu'elles feront nécessairement partie d'un groupe d'exécution quand le projet fonctionnera * qd ras*

*suivant a APRECISER par G3 si * il reste encore des composants non définis * qd ras*

*suivant sortie si * il n'y a plus de composants à définir ; on peut donc faire un bilan de la description de l'organisation par BILORGA * qd ras*

elem CREEX suivant a APRECISER par G3 si ras qd ras

*elem CREPE suivant c CREPE par G2 si * tant qu'il reste des personnes à identifier * qd ras*

*suivant c EDITPER * édition de la liste des personnes identifiées * par G2 si * cette liste est provisoirement close * qd ras*

*elem EDITPER suivant a DIFFPER * diffusion de la liste des personnes à tous les membres de G3 * par G2 si ras qd ras*

elem DIFFPER suivant a APRECISER par G3 si ras qd ras

fmajmo

*majmo MAMO * décomposition initiale de l'ensemble de l'application considérée comme un module **

*entrée c MAJMO par G2 si * la structure générale de l'application peut être déjà décrite * qd ras*

elem MAJMO suivant sortie si ras qd ras

fmajmo

On remarquera ici un exemple de module créé artificiellement (MAMO) qui ne contient qu'une commande (MAJMO), commande qui figure, par ailleurs, dans le module SPFOMO. Cette création de (MAMO) permet de distinguer les conditions d'activation différentes de la commande.

*majgr G2 * Le chef de projet est responsable de l'ensemble des études et de la réalisation. A ce titre, il contrôle, en particulier, toute la documentation produite par l'équipe de conception et de réalisation. Il est donc obligé d'assurer une fonction d'administrateur de la base de données mise en place avec MACSI-1.*

Il ne doit pas forcément être un spécialiste en informatique, car il est appuyé pour la réalisation technique par un responsable de la coordination informatique qui en assurera le suivi. Seule, une maîtrise parfaite de l'utilisation externe de MACSI-1 est exigée du chef de projet *

fmajgr

majgr G5 * *Coordinateur informatique du projet, il en suit la réalisation technique. Il est souhaitable qu'il connaisse le fonctionnement interne des programmes de MACSI-1, mais ceci n'est pas indispensable s'il existe quelqu'un d'autre de compétent dans ce domaine au sein du service informatique* *

fmajgr

Selon les projets, il se peut fort bien que soit la même personne, soit deux personnes différentes soient associées aux groupes G2 et G5. Nous n'avons par ailleurs rien dit sur leur rattachement à un service administratif. Dans le premier cas, le chef de projet est aussi coordinateur informatique ; dans le second, les fonctions sont réparties au niveau de deux individus distincts. On voit donc ici comment la notion de "groupe" dans MACSI-1 identifie bien des fonctions et non pas des personnes en tant que telles ni des services administratifs.

On pourrait continuer à décrire les différents éléments déclarés dans le module SPFOMO avec les commandes disponibles dans MACSI-1 : ce travail ayant déjà été fait sous une autre forme dans le chapitre 1 ne servirait ici à rien.
Il importe plutôt maintenant de voir quels tests de cohérence pourraient être réalisés de façon automatique sur les spécifications introduites dans la phase de description de l'organisation. Il faut donc préciser le contenu du module BILORGA.

creex BILORGA MAJMO MAJCO MAJDO MAJAC CRESE MAJGR

* *Un certain nombre de tests de contrôle peuvent être faits chaque fois qu'une des commandes de création ou de mise à jour d'un composant est utilisé, contrôles qui ne doivent pas être reportés au niveau de BILORGA. Ces contrôles devront donc être intégrés dans la logique de programmation des opérateurs associés à ces commandes.*

De façon générale, il existe différents types de contrôles qui, sous des modalités diverses, sont à prendre en compte pour toutes ces commandes.

1) *Pour une mise à jour, vérifier que le code du composant a déjà été déclaré.*
2) *Pour une création, vérifier que le code du composant n'a pas déjà été utilisé pour un autre.*
3) *Pour chaque commande, vérifier que la syntaxe de la règle de description qui lui est associée est respectée ; en particulier, vérifier que les propriétés et relations du composant, qui sont obligatoires dans la commande, sont bien spécifiées.*
4) *Vérifier que chaque module a au moins un point de sortie.*
5) *Interdire la récursivité dans un module.*

Les contrôles intégrés dans BILORGA porteront, par contre, sur les tests qui font intervenir plusieurs composants. Ce sera en particulier tous les contrôles qui mettent en jeu les cross-reference comme groupe-action, groupe-commande, module-action, module-commande, groupe-personne-service, etc ...

*

fcreex

majmo BILORGA

* *Contrôles de cohérence du projet qui font appel à l'ensemble des spécifications produites dans le cadre de SPFOMO. Ils sont tous destinés au chef de projet qui doit faire un diagnostic d'ensemble* *

entrée c BILPROV par G2

suivant sortie si * *il existe encore des composants qui sont spécifiés incomplètement. Par exemple, des propriétés*

*ayant la valeur "u", des commandes sans documents, des modules dont la structure n'est pas définie * qd ras*

*suivant c BILOR1 * liste des commandes et des actions qui apparaissent dans plusieurs modules * par G2*
*si * pour vérifier qu'il s'agit bien des mêmes opérations * qd ras*

*suivant c BILOR2 * Groupes ayant des personnes rattachées à des services différents * par G2*
*si * il y a dans ce cas risque de conflit d'attribution dans ces services * qd ras*

*suivant c BILOR3 * personnes rattachées à aucun groupe * par G2*
*si * leur identification est alors inutile * qd ras*

*suivant c BILOR4 * Pour chaque groupe liste des actions et commandes qu'il utilise * par G2*
*si * pour vérifier que ces tâches confiées au groupe représentent une fonction homogène * qd ras*

*suivant c BILOR5 * Groupe n'ayant que des commandes d'entrée de données **
*si * pour éviter que ces groupes ayant un rôle ingrat dans le projet ne se posent plus tard comme des O.S. de l'informatique * qd ras*

*suivant c BILOR6 * Commandes disponibles pour plusieurs groupes différents * par G2*
*si * pour vérifier qu'il n'y a pas erreur d'affectation * qd ras*

*suivant c BILOR7 * Commandes ou actions ayant des conditions d'exécution temporelles (clause qd) définies * par G2*
*si * pourvoir les contraintes de planning retenues dans le fonctionnement de l'application * qd ras*

```
elem BILOR1 suivant a BILAN * Bilan général effectué au vu des
     diagnostics * par G2
elem BILOR2 suivant a BILAN par G2
elem BILOR3 suivant a BILAN par G2
elem BILOR4 suivant a BILAN par G2
elem BILOR5 suivant a BILAN par G2
elem BILOR6 suivant a BILAN par G2
elem BILOR7 suivant a BILAN par G2
elem BILAN suivant sortie si ras qd ras
fmajmo
majmo REALISA
    * Implémentation de la version opérationnelle de l'application.
      Cette implémentation se fait généralement en deux étapes :
      1) Analyse organique et programmation
      2) Lancement
    *
  entrée m ANAPRO * Analyse organique et programmation *
  elem ANAPRO suivant m LANCER * Lancement de l'application *
              si * accord du chef de projet * qd ras
  elem LANCER suivant sortie si ras qd ras
fmajmo
```

L'implémentation du module ANAPRO sera l'objet du Chapitre 3.

3.a. Introduction

L'implémentation informatique doit suivre les spécifications fonctionnelles de l'application.
Il faut :

- transformer la structure logique des données soit en une base de données exploitable par des programmes spécialisés (Socrate par exemple),soit définir des fichiers exploitables par des programmes écrits en Cobol, Fortran, Assembleur, PL/1, etc ...,
- implémenter les commandes sous forme de programmes,
- tester ces programmes,
- répartir les tâches au sein de l'équipe.

On peut rappeler les problèmes essentiels liés aux spécifications techniques d'un produit informatique. Ces problèmes sont de deux ordres :

- D'une part le choix des moyens techniques les plus appropriés au niveau logiciel pour implémenter l'application,
- D'autre part, le choix des normes de description pour la logique des traitements et la structure physique des données.

Au niveau des moyens techniques, le problème principal qui se pose actuellement c'est l'écriture d'une application soit avec des fichiers gérés par des programmes écrits en Cobol ou PL/1, soit éventuellement depuis quelques années, l'utilisation des bases de données. La grande différence étant que d'un côté on manipule des fichiers, et de l'autre une structure de base de données, que le premier type d'application nécessite une programmation des entrées-sorties importante, mais permet peut-être des performances supérieures, le second type d'application permet une structure de données très complexe dont la gestion est assurée par le logiciel de base de données mais certainement au détriment des performances.

Un de nos objectifs est de pouvoir réaliser les spécifications techniques dans l'un comme dans l'autre des cas.

CHAPITRE 3

LE LANGAGE DE DESCRIPTION ORGANIQUE

En dehors du choix des logiciels, se pose le choix d'un formalisme pour décrire la logique des traitements informatiques. Plusieurs méthodes sont utilisées :

- Logique booléenne (méthode Corig)
- Tables de décisions
- Organigrammes classiques
- etc ....

Néanmoins, les efforts actuels dans le cadre de l'introduction à une programmation structurée conduisent de plus en plus à exiger une spécification de la logique des programmes sous forme de schémas de programmes.

Notre problème était de savoir quel type de formalisme choisir pour décrire des schémas de programme : nous aurions pu reprendre les propositions de Dijkstra (cf. [12]), ou bien les enseignements proposés par J. VOIRON, J. COURTIN et Y. CHIARAMELLA (cf. [8], [5]) à l'I.U.T.. Nous aurions pu aussi retenir la proposition de F. PECCOUD dans MACSI-3 (cf. [22]) où il définit un langage de spécification de la logique associé à la structure logique de données que nous avons évoquée au Chapitre 1. Mais dans MACSI-1, nous nous sommes limités au cas particulier, que nous connaissons bien, c'est-à-dire celui de l'étudiant ayant une expérience de Socrate.

Une pratique effective de Socrate permet de constater que la description d'une structure de données Socrate, tout en prenant en compte certains problèmes d'implémentation, laisse toujours apparaître une bonne partie de la logique du problème. D'autre part, le langage de requête Socrate a été conçu par J.R. ABRIAL pour être aussi proche que possible du calcul des prédicats et permettre une expression logique plus claire d'un problème (cf. [1]). On constate qu'une structure de données en Socrate et le source des macros-instructions ou des procédures qui opèrent dessus permettent, moyennant quelques commentaires, de prendre connaissance très rapidement de la logique d'une application.

Nous avons donc décidé de retenir,comme outil de spécification de la logique des opérateurs dans une application,le langage de requêtes Socrate. Pour que ce langage de requête puisse être utilisé, il nous faut donc associer à la structure sémantique des données que nous avons définie dans l'analyse fonctionnelle, une structure Socrate qui lui soit "isomorphe". Evidemment, il existe plusieurs structures possibles. On peut associer à la structure logique de données de nombreuses structures Socrate qui diffèrent au moins par les méthodes d'accès retenues. Il faut, cependant, s'assurer que la structure Socrate choisie est compatible avec la structure logique de données et donc vérifier en particulier le respect des règles suivantes :

1) Aux ensembles de base correspondent les entités de plus haut niveau dans la structure Socrate.

2) Les noms d'entités correspondent aux codes des ensembles (CODENS) associés.

3) Le nombre maximum des réalisations d'une entité de plus haut niveau doit être égal à la taille de l'ensemble (TAILLENS).

4) Les caractéristiques simples de ces entités correspondent aux propriétés des ensembles de base (PROPRIETES-ENS). Le type de chaque caractéristique simple doit être cohérent avec celui de la propriété correspondante.

5) Les propriétés considérées comme des clés d'un ensemble de base (CLEFS-ENS) seront généralement des caractéristiques discriminantes de l'entité Socrate associée.

6) Pour implémenter les relations entre les ensembles, on ne peut pas avoir la même rigueur car de nombreuses combinaisons d'implémentation sont alors possibles.

Prenons l'exemple suivant volontairement simplifié, où nous avons 2 ensembles et une relation :

E_1 = Professeurs	E_2 = Matières	R_1 = Enseignement
Taille de E_1 = 50	Taille de E_2 = 20	Taille de R_1 = 50
P_{11} = Nom	P_{21} = Titre	ARG-1 = Professeurs
		ACCES-1 = 'Spécialité'
		MAX-1 = 1
		ARG-2 = Matières
		ACCES-2 = 'Enseignant'
		MAX-2 = 1

Donnons quelques solutions pour implémenter cette structure logique de données en Socrate :

```
Entité 50 PROFESSEUR
 Début
      NOM mot discr
      SPECIALITE réfère une MATIERE
 fin
```

```
Entité 20 MATIERE
 Début
      TITRE mot discr
      ENSEIGNANT réfère un PROFESSEUR
 fin
```

Solution a

```
Entité 50 PROFESSEUR
 Début
      NOM mot discr
      SPECIALITE mot
 fin
```

```
Entité 20 MATIERE
 Début
      TITRE mot discr
      ENSEIGNANT réfère un PROFESSEUR
 fin
```

Solution b

```
Entité 50 PROFESSEUR
 Début
      NOM mot discr
      SPECIALITE de 1 à 20
 fin
```

```
Entité 20 MATIERE
 Début
      TITRE mot discr
      ENSEIGNANT réfère un PROFESSEUR
 fin
```

Solution c

Neuf solutions différentes peuvent être déduites en faisant combiner les 3 solutions ci-dessus pour la seule implémentation de "ENSEIGNEMENT" par ses deux fonctions d'accès dans PROFESSEUR et dans MATIERE.

Il existe d'autres solutions possibles comme le cas suivant :

Entité 50 PROFESSEUR
 Début
  NOM mot discr
  SPECIALITE réfère un ENSEIGNEMENT
 fin

Entité 20 MATIERE
 Début
  TITRE mot discr
  ENSEIGNANT réfère un ENSEIGNEMENT
 fin

Entité 50 ENSEIGNEMENT
 Début
  PROF réfère un PROFESSEUR
  MAT réfère une MATIERE
 fin

Solution x

On peut dégager quelques règles :

6-1) Il existe dans le cas où le maximum (MAX) de la fonction d'accès d'une relation est égal à 1 trois méthodes d'accès disponibles :

- par un pointeur (refère) (cf. solution a = SPECIALITE et ENSEIGNANT, solutions b et c = ENSEIGNANT).
- par identification de l'élément en relation avec la valeur de son code discriminant. Ainsi, dans la solution b, SPECIALITE a pour valeur la valeur du TITRE de la MATIERE associée.
- par identification de l'élément en relation avec le numéro d'ordre dans la suite des réalisations de l'entité (solution c = SPECIALITE).

6-2) Si le maximum de la fonction d'accès d'une relation est supérieur à 1, alors :

- on peut utiliser une entité de pointeurs ayant comme nom celui de la fonction d'accès et comme taille celle du maximum de la fonction d'accès.
- on peut utiliser la méthode d'accès par fichier inversé (chaîne d'inverse de Socrate ayant pour nom et pour taille ceux de la fonction d'accès).

6-3) Dans les deux cas, la solution "x" peut être envisagée. Il est plus naturel d'utiliser cette implémentation dans le cas où l'ensemble-relation a lui-même des propriétés (PROPRIETES-REL).

Le choix de la structure Socrate compatible avec la structure logique des données dépend essentiellement des traitements envisagés par les commandes demandées dans la phase d'analyse fonctionnelle . D'autres critères peuvent influencer ce choix :

- Structure plus modulaire (solution x) qui permet des citations plus courtes ,
- L'occupation de place dans l'espace physique peut déterminer le choix entre une entité de références et une chaîne d'inverses,
- Suppression automatique des pointeurs par référence vers une entité lorsque celle-ci est supprimée,
- Possibilité de modification de la structure.

Ce choix est donc un choix non automatisable dans l'état actuel de nos compétences ; aussi est-il confié au responsable informatique du projet.

Cette structure étant définie, les analystes-programmeurs devront définir les différentes parties de la logique de l'application sous forme de procédures ou de macro-instructions Socrate. Un des avantages de ce choix est de permettre éventuellement au langage de description de la logique d'être exécutable, problème que nous avons rencontré lorsque, pour une application, nous avons jugé utile de construire préalablement un prototype de cette application. Entendons par prototype d'une application une implémentation informatique rapidement réalisée et fonctionnellement équivalente qui nous permet de faire une simulation de ce qui serait l'application si elle devenait opérationnelle (cf. [22]).

Les futurs utilisateurs jugeront sur pièce et pourront exprimer des critiques de façon à modifier le prototype. L'une des qualités principales du prototype est donc de pouvoir être facilement modifiable.

Nous allons maintenant décrire les différentes étapes que nous avons retenues et les commandes disponibles dans MACSI-1 pour faire cette analyse organique. Par souci d'homogénéité, nous utiliserons le même formalisme qu'au chapitre 2 (structure modulaire) pour spécifier les étapes et la même représentation graphique (chapitre 1) pour décrire les commandes.

Nous ne sommes pas dupes du fait que ce genre de graphique n'a qu'un aspect visuel et ne pourrait pas être "stocké en machine". Nous indiquerons au chapitre 4 (cf. § 4.e.) comment ces graphiques seront traduits en automates d'états finis et transformés en règles grammaticales LACSYST pour qu'une telle description des composants MACSI-1 (actions, modules, commandes, ...) soit stockable.

3.b. Description des étapes de l'analyse organique = *anapro*

Nous allons reprendre le module ANAPRO déjà cité dans le chapitre 2 pour le décomposer en ses éléments, de façon à décrire l'enchaînement des étapes de l'analyse organique et de la programmation des opérateurs.

majmo ANAPRO

* *Ce module regroupe les opérations relatives à ce qu'on appelle couramment l'analyse organique et la programmation. En effet, une fois que la description fonctionnelle de l'application est complètement définie et que ces spécifications sont éditées, les étapes à suivre pour implémenter une version opérationnelle sont les suivantes :*

1) *Détermination d'une structure physique des données*
2) *Répartition des tâches de programmation parmi les différents membres de l'équipe de programmation*
3) *Etude et analyse des opérateurs à programmer pour déterminer leur logique*
4) *Programmation des opérateurs*
5) *Contrôle des opérateurs programmés avec jeux d'essais significatifs* *

*entrée c EDICO * Edition des commandes à implémenter **
par auto
*suivant m STRDO * Choix des structures physiques des données **
*si ras qd * avec un délai d'environ 1 mois au plus **
*elem STRDO suivant m PLN * Planning de travail **
si ras qd ras
*elem PLN suivant m ANAOP * Analyse des opérateurs **
*si ras qd * au plus tôt et en fonction des délais demandés **
*elem ANAOP suivant m PROGR * Programmation des opérateurs **
*si * pour les opérateurs analysés * qd ras*
*elem PROGR suivant c MAJEG * description complète d'un enregistrement de fichier ou de document * par G5 si * il reste un enregistrement incomplet * qd ras*
*suivant m CTL * Contrôle des opérateurs programmés **
*si * pour les opérateurs programmés tous les enregistrements ont été décrits par majeg * qd ras*

elem MAJEG *suivant c* MAJEG *par* G5 *si* * *il reste un enregistrement incomplet* * *qd ras*
suivant m CTL *si* * *tous les enregistrements ont été complètement définis* * *qd ras*
elem CTL *suivant c* RECEP * *Réception des opérateurs implémentés* *
par G5 * *Responsable informatique* *
si * *pour les opérateurs avec contrôle satisfaisant* *
qd * *après la date prévue d'implémentation de ces opérateurs* *
suivant m PROGR
si * *Pour les opérateurs non satisfaisant le contrôle* *
qd ras
elem RECEPT *suivant sortie si ras qd ras*
fmajmo

3.c. Implémentation physique des données = *strdo*

majmo STRDO

* *Le responsable informatique ayant fait le choix des moyens techniques pour implémenter la version opérationnelle doit déduire une structure physique des données. En faisant référence à la figure 3-1, nous voyons qu'en tout état de cause une structure Socrate doit être implémentée à partir de la structure logique des données pour permettre par la suite de faire la description de la logique des traitements des opérateurs en se référant à cette structure. Alors, trois cas peuvent se présenter :*

1) *L'application opérationnelle est elle-même écrite en Socrate, ce qui nous permet de conserver cette structure de données pour exprimer la logique et éventuellement la programmation des opérateurs.*

2) *L'application opérationnelle est écrite en Cobol, Fortran, PL/1 ou autres, alors il nous faut définir en plus de la structure Socrate, les fichiers sur lesquels ces programmes agiront.*

3) *L'application opérationnelle est une version mixte des deux possibilités pour laquelle il faut aussi définir des fichiers.*

A ce niveau, on définit des fichiers par une description des caractéristiques globales, on donne un nom à son enregistrement associé, ce qui permet de faire l'organisation des principales unités de traitement. Ce n'est que plus tard que l'on définira le contenu des enregistrements. Il nous faut aussi associer un enregistrement aux documents décrits qualitativement dans la phase fonctionnelle *

*entrée m CHOIX * Etude et choix d'une structure Socrate *
suivant c CREST * Enregistrement de la structure Socrate *
par G5 si * dans le module CHOIX une nouvelle structure est à enregistrer * qd ras
suivant c BONST * Validation d'une structure *
par G5 si ras qd ras
elem CREST suivant c BONST par G5 si ras qd ras
elem BONST suivant c CREFI * Création d'un fichier *
par G5 si * il y a des fichiers à créer * qd ras
suivant c EDIDO * Edition des documents *
par G5 si ras qd ras
suivant sortie si * il n'y a plus ni fichier à créer ni enregistrements à associer aux documents * qd ras
elem CREFI suivant c CREFI par G5 si * il y a d'autres fichiers à créer * qd ras
suivant sortie si ras qd ras
elem EDIDO suivant c MAJEGDO * Association d'un enregistrement à un document *
par G5 si * il y a des documents auxquels il faut associer un enregistrement * qd ras
elem MAJEGDO suivant c MAJEGDO par G5
si * il y a d'autres documents auxquels il faut associer un enregistrement * qd ras
suivant sortie si ras qd ras
fmajmo*

3.c.1. Enregistrement de la structure de base de données = *crest, bonst*

majco CREST

* *Cette commande permet d'enregistrer le source d'une structure Socrate. Le graphique associé est le suivant :*

crest CODESTR LIBELLE * source * SOURCE * fcrest*

*Le mot-clé crest permet d'identifier la commande de création d'une structure Socrate. On donne le code de la structure (CODESTR) suivi du libellé (LIBELSTR), du mot-clé source, de la spécification du source de la structure Socrate associée (SOURCESTR). Pour terminer la création de la structure, on donne le mot réservé fcrest **

type 1

*doc D21 * Bordereaux de perforation ou clavier de télétype **

fmajco

majco BONST

* *Cette commande permet de valider la structure Socrate par rapport à laquelle nous décrirons les opérateurs*
Le graphique associé est le suivant :

bonst CODESTR fbonst

*
type 1
doc D21

fmajco

3.c.2. Association d'un enregistrement à chaque document = *majegdo*

majco MAJEGDO

* *C'est dans cette phase organique que les informaticiens vont associer un enregistrement à tout document à partir de sa définition. En effet, dans la phase fonctionnelle, les documents sont définis par les analystes (R.D. de mise à jour de documents majdo) mais le contenu est précisé uniquement sous forme d'un texte libre dans la définition (DEFDO). Les informaticiens associeront un enregistrement au document et spécifieront la propriété clé (simple ou composée). La commande de mise à jour de l'enregistrement du document est la suivante :*

majegdo CODEDO enr CODENR ⟨ * *LIBELLE* * ⟩ *clé CODEPP* → *fmajegdo*

Le mot-clé majegdo identifie cette commande. Le code du document à mettre à jour est spécifié (CODEDO) suivi du mot-clé enr qui permet de spécifier l'enregistrement associé. On donne le code de l'enregistrement (CODENR) et si celui-ci n'a pas été déclaré par une autre commande, on donne son libellé (LIBELENR). Si l'on veut associer une propriété clé au document, on donne le mot réservé clé suivi du code de la propriété (CODEPP). Le rattachement d'un enregistrement au document se termine par le mot-clé fmajegdo *

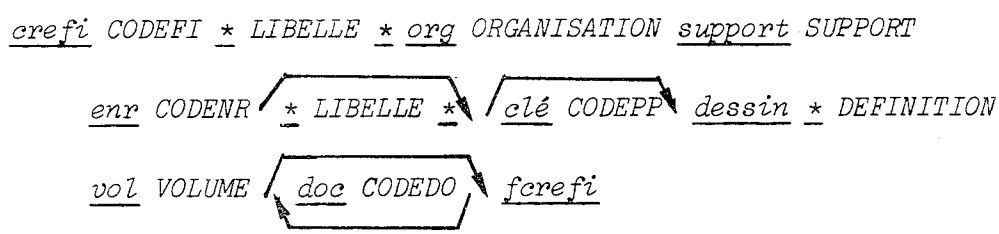
type 1
doc D21

fmajco

3.c.3. Création des fichiers = *crefi*

majco CREFI

* *Cette commande permet la spécification des caractéristiques globales d'un fichier. Cette description se fait conformément au graphique suivant :*



Le mot-clé crefi permet d'identifier la commande de création des fichiers. Il sera suivi du code du fichier (CODEFI) et de son libellé (LIBELFI). Ensuite, l'organisation du fichier sera spécifiée par le mot-clé org suivi de :

1 pour un fichier en organisation séquentielle
2 pour un fichier en organisation séquentielle indexée
3 pour un fichier en organisation directe.

Le support du fichier sera défini par le mot réservé support suivi de :

1 pour fichier sur bande magnétique
2 pour fichier sur disque magnétique
3 pour fichier sur cartes perforées
4 pour fichier sur ruban perforé.

L'enregistrement du fichier sera déclaré par le mot-clé enr, le code CODENR et le libellé de l'enregistrement (LIBELENR) dans le cas où l'enregistrement n'a pas été déclaré préalablement.
Si le fichier a une clé (propriété simple ou composée) permettant un ordre de rangement des articles du fichier (CLESFI)), elle sera spécifiée par le mot réservé clé suivi du code de la propriété clé (CODEPP). Pour continuer la spécification du fichier, l'analyste donne sa définition par le mot-clé dessin suivi du texte explicatif, le volume par le mot-clé vol suivi du chiffre correspondant au nombre maximum de caractères prévus du fichier et la liste de documents associés (DOCASS) par la répétition du mot-clé doc suivi du code du document (CODEDO) autant de fois qu'il y a de documents à spécifier.

Finalement le mot-clé fcrefi *signale la fin de la description du fichier.*

REMARQUES :

(1) *L'enregistrement déclaré par le fichier sera l'objet d'une mise à jour ultérieure. On définira en particulier les propriétés concernées et spécifiées dans la définition du fichier.*

(2) *Les documents associés aux fichiers doivent être complètement définis ou au moins déclarés lors de la description des commandes*

```
*
type 1
doc D21

fmajco
```

3.d. Spécification des opérateurs

3.d.1. Etapes et moyens de l'implémentation

Le processus de spécification des opérateurs est décomposé en plusieurs étapes. Pour un opérateur, qui est l'implémentation informatique d'un traitement exprimé en langue naturelle dans une commande, on prévoit les états suivants :

1) Une définition qualitative (dans DEFOP) déduit par le coordinateur informaticien à partir de la définition de la commande.

2) Une description de sa logique (dans SPECIF) suite à une analyse de l'opérateur par le programmeur.

3) Une définition de son source (dans SOURCE) suite à la programmation de l'opérateur.

Donc, les opérateurs passent par une étape d'analyse qui aboutira à la détermination de sa logique, ensuite cette logique sera programmée soit dans un langage de gestion de base de données type Socrate, soit dans un langage type COBOL, GAP, PL/1 ou autres. Finalement, un contrôle s'impose pour tester le bon fonctionnement de l'opérateur.

Continuons avec la description des éléments du module ANAPRO (cf. p. 2.17) qui décrit cet enchaînement.

majco EDICO * Edition de toutes les commandes complètement spécifiées et disponibles à l'implémentation. Autrement dit, édition des commandes qui sont dans l'état 1 *

type 2

doc D103 * Etat des commandes à implémenter *

fmajco

majmo PLN

* Ce module correspond à la répartition des commandes à implémenter après étude par le responsable informatique. En effet, avec le document des commandes à implémenter, le responsable informatique fera l'étude de ces commandes, choisira celles qui sont à implémenter et distribuera la responsabilité d'implémentation entre ses analystes programmeurs. Il distribuera aussi la responsabilité d'implémentation des opérateurs de service qu'il envisage. Finalement, une édition du planning sera exécutée *

entrée m ETU * Etude des commandes et opérateurs de service à implémenter *

suivant c LANCE * Lancement d'un opérateur *

par G5 si * Il y a une commande à implémenter *

qd ras

suivant c CREOP * Création d'un opérateur de service *

par G5 si * Il y a un opérateur de service à implémenter *

qd ras

suivant sortie si * Aucune commande ou aucun opérateur de service n'est à implémenter *

elem LANCE suivant c LANCE par G5 si * Il y a d'autres commandes à implémenter *

qd ras

suivant c ED-PLAN * Edition du planning de travail * par G5

si * Il n'y a plus de commandes à distribuer *

qd ras

elem CREOP suivant c CREOP par G5 si * Il y a d'autres opérateurs de service à implémenter *

qd ras

suivant c ED-PLAN par G5 si * Il n'y a plus d'opérateurs de service à implémenter *

qd ras

elem ED-PLAN suivant sortie si ras qd ras

fmajmo

majmo ANAOP

** Les analystes-programmeurs sachant quels sont leurs commandes et opérateurs de service à implémenter vont faire pour chacun d'entre eux l'étude des spécifications pour aboutir à la détermination de leur logique sous forme de schémas de programmes **

*entrée m ETUCO * Etude des commandes et opérateurs à implémenter **
*suivant c ANALYSE * Spécifications logiques de l'opérateur **
*par G6 * Analystes-programmeurs * si * l'analyste est en mesure de donner les spécifications logiques de l'opérateur * qd ras*
*suivant m ETA2-3 * Changement à l'état 2 ou 3 de la commande **
*si * l'analyste manque de précisions sur la commande ou il la considère impossible à implémenter * qd ras*
*suivant sortie si * il n'y a plus de commandes ou d'opérateurs à analyser * qd ras*
elem ANALYSE suivant m ETUCO si ras qd ras
elem ETA2-3 suivant m ETUCO si ras qd ras
fmajmo

majmo PROGR

* Les analystes-programmeurs feront la programmation des opérateurs suivant la logique qu'ils ont eux-mêmes spécifiée lors de l'analyse. Une étape d'étude est suivie de la commande PROG qui permet la spécification du texte source associé à l'opérateur. Etant donné que l'implémentation d'une commande ou d'un opérateur de service sous forme de programme peut entraîner l'utilisation des fichiers de travail, une commande de mise à jour des fichiers majfi sera disponible *

entrée m ETULO * Etude des opérateurs à programmer *
suivant c PROG * Spécification du texte source *
par G6
si * l'analyste peut donner la spécification du texte source de l'opérateur * qd ras
suivant c MAJFI * Mise à jour des fichiers de travail *
par G6
si * des fichiers déclarés comme fichiers de travail d'un opérateur sont à spécifier * qd ras
suivant a DEMOS * Demande auprès du responsable informatique d'un opérateur de service *
par G6
si * l'analyste le considère nécessaire pour la programmation de l'opérateur * qd ras
suivant sortie si * il n'y a plus d'opérateurs ou de fichiers de travail à spécifier * qd ras
elem PROG suivant c MAJFI par G6
si * il y a des fichiers de travail déclarés par PROGRAMME *
qd ras
suivant m ETULO si ras qd ras
elem MAJFI suivant m ETULO si ras qd ras
elem DEMOS suivant m ETULO si ras qd ras
fmajmo

```
majmo CTL
    * Une fois les opérateurs programmés, il faut les contrôler. Le
      contrôle, s'il est satisfaisant, bascule à 4 l'état de cet opérateur.
      L'état 4 d'un opérateur indique qu'il est programmé, que tous les
      fichiers qu'il utilise sont définis, que tous les opérateurs qu'il
      appelle sont eux aussi dans l'état 4 et que l'exécution avec un jeu
      d'essai significatif est satisfaisante. Un message de wargning ou
      d'erreur, selon le cas, sera édité pour les opérateurs qui ne
      satisfont pas le contrôle *
  entrée m CONTROLE * Contrôle d'un opérateur *
        suivant m TEST * Contrôle par un jeu d'essai *
         si * le contrôle est satisfaisant * qd ras
        suivant sortie si * plus d'opérateurs à contrôler * qd ras
  elem TEST suivant m CONTROLE si ras qd ras
fmajmo
```

majco RECEPT

* *Cette commande est activée par le mot réservé recept. Elle examine toutes les commandes en cours d'implémentation (commandes dans l'état 4). Pour celles dont l'opérateur principal associé est programmé et contrôlé (opérateur principal dans l'état 4) et que la date prévue de fin d'analyse et programmation (DATE-2-CO) est inférieure ou égale à la date du jour :*

1) L'état de l'opérateur principal est basculé à 5 (livré)
2) La DATE-3-CO de la commande est mise à jour avec la date du jour
3) L'état de la commande passe à 5
4) Edition du code de la commande et de l'opérateur principal associé.

Elle fait aussi l'édition des commandes dont le délai prévu de fin d'analyse et programmation arrive à terme mais dont l'opérateur principal associé n'est pas encore programmé et contrôlé. En particulier, pour chacune de ces commandes, on sort le code, le code de l'opérateur principal associé et le nom de l'analyste-programmeur responsable de l'implémentation de la commande *

type 2

doc D21

fmajco

majco LANCE

* *Cette commande est utilisée par le responsable de l'implémentation informatique des opérateurs. Elle permet à celui-ci de déclarer un opérateur principal associé à la commande qu'il désire voir implémenter et de spécifier un analyste-programmeur responsable de son implémentation. Il indiquera également la date de livraison désirée. Le responsable informatique en activant plusieurs fois la commande LANCE, établit le "planning" concernant la programmation des commandes.*
Le graphe correspondant à l'introduction des spécifications est le suivant :

*lance CODECO CODEOP * LIBELLE * def * DEFINITION * par CODEPER pour DATE flance*

L'activation de l'opérateur qui permet le lancement de l'implémentation d'une commande se fait par le mot-clé lance. Il sera suivi du code de la commande (CODECO) et du code que l'on veut donner à l'opérateur associé (CODEOP). Ensuite le libellé de l'opérateur (LIBELOP) puis le mot-clé def pour spécifier la définition de l'opérateur (DEFOP). Cette définition est l'explication informatique de la définition de la commande donnée dans la phase fonctionnelle. Ensuite, un analyste-programmeur est affecté pour l'étude de l'implémentation de l'opérateur. Pour cela, on donne le mot-clé par suivi du code de l'analyste-programmeur (CODEPER). Finalement, le mot-clé pour suivi de la date désirée de livraison permet de spécifier le délai d'analyse et de programmation de l'opérateur. L'opérateur de lancement se termine par le mot-clé flance.

REMARQUES : Il y a un certain nombre de mises à jour qui s'exécutent automatiquement lors de l'exécution de LANCE

(1) *La caractéristique OPERATEUR-ASSOCIE de la commande est mise à jour avec l'opérateur que l'on vient de déclarer.*

(2) *La caractéristique COM-ASSOCIE de l'opérateur déclaré par LANCE est mise à jour avec la commande citée dans LANCE (CODECO).*

(3) *La date prévue de fin d'analyse et programmation de la commande (DATE-2) est mise à jour avec la date spécifiée dans LANCE comme date prévue de livraison de l'opérateur.*

(4) *L'usage de l'opérateur (USAGE) est définie comme OPERATEUR PRINCIPAL.*

(5) *L'état de la commande (ETATCO) passe de l'état 1 à l'état 4.*

(6) *L'état de l'opérateur (ETATOP) est initialisé à 1* *

type 1

doc D15 * *Le document ayant la description graphique du "lancement" d'un opérateur* *

fmajco

majco CREOP

* *Cette commande permet au responsable informatique de déclarer un opérateur de service pour qu'il soit implémenté. Il spécifiera en particulier sa définition et le responsable de l'analyse et de la programmation.*
Le graphique associé à cette commande est le suivant :

*creop CODEOP * LIBELLE * def * DEFINITION * par CODEPER pour DATE fcreop*

Le mot-clé creop est suivi du code de l'opérateur (CODEOP), du libellé (LIBELOP), du mot-clé def pour spécifier la définition de l'opérateur (DEFOP) sous forme de texte. Le responsable de l'implémentat on est spécifié par le mot-clé par suivi du code de la personne (CODEPER). La DATE de livraison suit le mot-clé pour. Le mot-clé fcreop identifie la fin de la création de l'opérateur de service.

REMARQUES

(1) *L'état de l'opérateur est automatiquement mis à 1.*

(2) *La propriété USAGE de l'opérateur devient automatiquement égal à 3 (opérateur de service).*

(3) *Cet opérateur fera l'objet ensuite d'une analyse et d'une programmation **

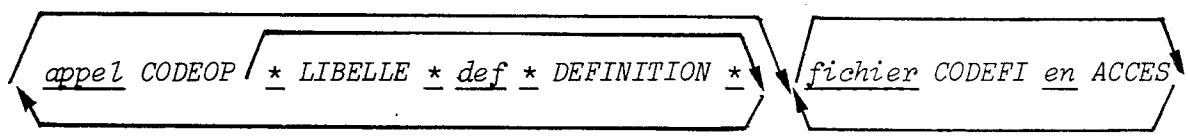
type 1
doc D21

fmajco

majco ANALYSE

* *Cette commande permet à l'analyste-programmeur de spécifier la logique du traitement d'un opérateur sous forme de schémas de programme. Voyons les différentes spécifications qui sont précisées dans cette étape et l'ordre d'entrée de ces spécifications :*

analyse CODEOP type TYPE accès ACCES mode MODE



*spécif * SPECIFICATION * fanalyse*

La commande d'analyse est activée par le mot-clé analyse suivi du code de l'opérateur à implémenter. Il nous permet de spécifier le type de l'opérateur (TYPOP) par le mot-clé type suivi du code numérique correspondant au type. L'accès a la base de l'opérateur (ACCES-OP) par le mot-clé accès suivi du code numérique correspondant à l'accès. Ensuite, le mode d'utilisation par rapport au traitement est introduit par le mot-clé mode suivi du code numérique correspondant au mode. Si l'opérateur est décomposé en plusieurs opérateurs plus élémentaires ou bien s'il fait appel à d'autres opérateurs, on spécifie la liste de ces opérateurs secondaires. Pour chaque opérateur secondaire, on donne le mot-clé appel et le code de l'opérateur (CODEOP). Eventuellement, si l'opérateur n'a pas encore été déclaré, on donne le libellé (LIBELOP) et la définition (DEFOP) par le mot-clé def suivi de la définition. Cette opération est répétée pour tous les opérateurs appelés. S'il n'y a pas d'opérateurs appelés ou plus élémentaires, après la spécification du mode, on passe directement à l'indication des fichiers de bases manipulés directement par l'opérateur. Le mot-clé fichier sera suivi du code du fichier (CODEFI) et l'accès au fichier (ACCES-FICH) sera indiqué sous forme d'un code numérique après le mot-clé en. On répétera cette opération de déclaration pour chacun des fichiers manipulés. Enfin, l'analyste-programmeur devra spécifier la logique de l'opérateur, généralement sous forme de schémas de programme, en donnant après le mot-clé spécif les spécifications fonctionnelles de l'opérateur (SPECIF). Le mot-clé fanalyse permet d'identifier la fin de la commande ANALYSE.

REMARQUES

(1) *L'utilisation de la commande ANALYSE ne peut se faire que pour spécifier des opérateurs qui ont été déjà lancés par lance, ou bien déclarés par analyse comme opérateur secondaire, ou encore pour les opérateurs de service déclarés par creop.*

(2) *Le type de l'opérateur (TYPOP) est déterminé de la façon suivante :*

1 = Saisie
2 = Contrôle
4 = Traitement
8 = Edition.

Une combinaison quelconque est représentée par l'addition des types élémentaires.

Exemple : 3 = Saisie-contrôle
13 = Saisie-traitement-édition.

(3) *L'accès à la base est représenté par :*

1 = accède en lecture
2 = accède en écriture
3 = accède en lecture-écriture
4 = n'accède pas à la base.

(4) *Le mode d'utilisation vis à vis du traitement est représenté par :*

1 = conversationnel
2 = batch
3 = batch ou conversationnel.

(5) *La citation des opérateurs secondaires n'ayant pas été déclarés auparavant par lance ou par analyse permet de spécifier leurs caractéristiques suivantes : CODEOP, LIBELOP, DEFOP, USAGE = 2, ETATOP = 1, OP-APPELANT, REALISATEUR qui est le même analyste-programmeur. Pour l'opérateur appelant, la caractéristique OP-APPELE est mise à jour.*

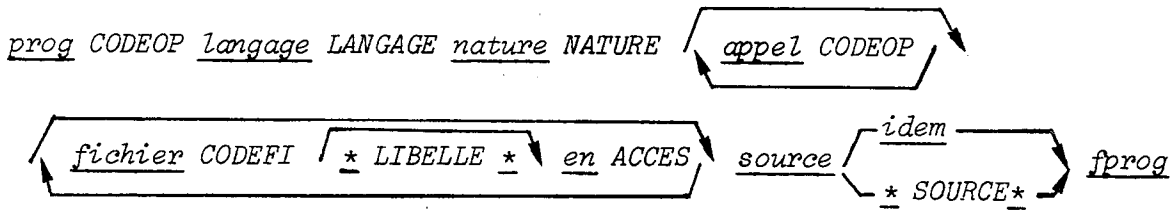
(6) *Tous les opérateurs secondaires ainsi définis devront faire l'objet d'une spécification ultérieure par analyse et par prog **

type 1
doc D21

fmajco

majco PROG

* *Cette commande permet à l'analyste-programmeur de spécifier le texte correspondant à l'implémentation de la logique de l'opérateur dans un langage de programmation.*
Le graphe associé à cette commande est le suivant :



Le mot-clé prog permet l'activation de cette commande suivi du code de l'opérateur à spécifier (CODEOP). L'indication du langage de programmation (LANGOP) se fait par le mot-clé langage suivi du code associé au langage de programmation, par exemple :

1 *Socrate*
2 *Assembleur*
3 *Cobol*
4 *Fortran*
5 *JCL.*

La nature de l'opérateur par le mot réservé nature suivi de :

1 *pour programme*
2 *pour macro-instructions.*

L'opérateur peut faire éventuellement appel à un ou plusieurs opérateurs de service : par exemple, des opérateurs de compactage d'une donnée, des opérateurs des contrôles standards sur une donnée, des opérateurs d'éditions, etc. Si c'est le cas, la citation des opérateurs de service consiste à répéter le mot-clé appel suivi du code de l'opérateur appelé (CODEOP) autant de fois que nécessaire. On peut avoir à définir des fichiers de travail propres à l'opérateur pour des opérations de stockages temporaires. Ces fichiers n'ont de durée de vie que celle de l'opérateur. Ils sont pratiques pour la programmation et ne sont généralement pas perçu au niveau fonctionnel : des fichiers intermédiaires, des fichiers de manoeuvre pour les tris, etc ...
On donne le mot-clé fichier suivi du code du fichier (CODEFI). Si ce fichier a déjà été déclaré ou défini, il suffit simplement de rappeler son code, dans le cas contraire on donnera son libellé (LIBELFI).

L'accès au fichier (ACCES-FICH) est spécifié par le mot-clé en *suivi de :*

1 pour lecture
2 pour écriture
3 pour lecture-écriture.

Enfin, il faut spécifier le texte correspondant au programme. Le mot-clé source *sera suivi soit du texte source, soit par le mot réservé* idem *pour indiquer que le texte source est le même que le texte correspondant aux spécifications logiques de l'opérateur. Ce cas se présente lorsque les spécifications logiques sont faites dans un langage qui est aussi exécutable. Le mot-clé* fprog *indique la fin de l'opérateur* prog ***

type 1
doc D21

fmajco

majco MAJFI

* Cette commande permet la mise à jour des fichiers de travail initialisés par la commande PROG.
Le schéma de description associé est le suivant :

majfi CODEFI org ORGANISATION support SUPPORT enr CODENR

* LIBELLE * | clé CODEPP | dessin * DEFINITION * vol VOLUME fmajfi

Le processus pour la mise à jour d'un fichier est le même que celui de la création d'un fichier, mais :

1) On ne spécifie pas le libellé du fichier car il s'agit d'une mise à jour.
2) Les fichiers de travail n'ont pas de documents associés *

type 1
doc D21

fmajco

3.d.2. Remarques méthodologiques

(1) On aurait pu prendre l'option d'imposer une analyse strictement descendante des opérateurs, c'est-à-dire que si l'on analyse un opérateur A (étape k-ième) et que celui-ci appelle de nouveaux opérateurs B_i, à la fin de l'analyse de l'opérateur A l'étape suivante (k+1-ième) consiste à faire l'analyse de tous les opérateurs B_i appelés. Pour la programmation d'un opérateur, la démarche serait le contraire, c'est-à-dire strictement ascendante. On devrait commencer la programmation des opérateurs appelés qui sont au dernier niveau et par étapes successives remonter jusqu'à la programmation de l'opérateur A. A chaque étape, les fichiers de travail initialisés devraient être complètement spécifiés.

L'analyse descendante des opérateurs permet :

- une homogénéité dans le degré de finesse des opérateurs secondaires
- l'établissement de normes de programmation
- une vision globale des besoins en opérateurs de service.

La programmation ascendante pourrait être imposée si l'on interdit par programme l'emploi de la commande "prog" pour un opérateur qui n'a pas tous ses opérateurs appelés (secondaires ou de service) déjà programmés. C'est-à-dire lors d'une programmation (prog) pour un opérateur A, on contrôlerait automatiquement que tous les sous-opérateurs B_i indiqués dans la phase d'analyse ont fait l'objet d'une programmation aussi par prog.

L'analyse strictement descendante et la programmation strictement ascendante ne sont pas directement contrôlées par MACSI-1, néanmoins elles sont fortement conseillées.

(2) Le module qu'on a appelé CONTROLE (cf. p. 3.19) devait normalement être une commande. En effet, cette commande vérifierait pour un opérateur donné :

1°) qu'il est dans l'état 3, c'est-à-dire programmé

2°) que tous les fichiers qu'il appelle sont complètement définis

3°) que tous les opérateurs qu'il appelle sont dans l'état 4 ou peuvent passer dans l'état 4, c'est-à-dire que cette commande CONTROLE ferait une étude descendante des opérateurs appelés jusqu'au dernier niveau et contrôlerait à chaque niveau l'état des opérateurs appelés.
Pour un opérateur, deux cas possibles se présentent à la fin de cette opération :

1) le contrôle est satisfaisant, donc il est prêt pour être soumis au test du jeu d'essai

2) sinon un diagnostic d'erreur sera signalé pour cet opérateur, et on éditera une liste complète de tous ses sous-opérateurs qui ne satisfont pas le contrôle. Pour chacun on indiquerait les motifs de l'échec.

(3) Le lecteur a pu constater que pour les opérateurs qui travaillent directement sur la structure de base de données la liaison avec la structure logique des données (ESPDONNEE) n'est pas explicitement donnée par l'analyste programmeur chargé des spécifications.
Il serait souhaitable que cette liaison soit faite automatiquement. En effet, dans la commande PROG,une fois qu'un tel opérateur est spécifié, un programme d'analyse du texte SOURCE pourrait faire la liaison explicite entre les données utilisées dans le programme et les propriétés de la structure logique de données. S'il y a des identificateurs ne correspondant pas aux propriétés dans la structure logique des données, ils seront considérés comme des variables faisant partie d'un ensemble particulier de travail (TYPENS = 10).

3.e. Définition de la structure des enregistrements = *majeg*

C'est à partir soit de la définition d'un document (*DEFDO*), soit de la définition d'un fichier (*DESSIN-ENR*) que les analystes décriront les propriétés composant leurs enregistrements. Rappelons-nous que ces enregistrements ont été déclarés par les commandes ***CREFI*** (création de fichier), *MAJFI* (mise à jour de fichier) et *MAJEGDO* (rattachement d'un enregistrement à un document).

La commande *MAJEG* permet la mise à jour des enregistrements initialisés. Le graphique correspondant est le suivant :

majeg *CODENR* * *DEFINITION* * (*prop* *CODEPP*) *fmajeg*

Pour faire la définition complète d'un enregistrement, l'analyste donne le mot-clé *majeg* suivi du code de l'enregistrement à spécifier (*CODENR*) et de sa définition (*DESSIN*).

La liste des propriétés de l'enregistrement (*ESPDONNEE*) est spécifiée par la répétition du mot-clé *prop* suivi du code de la propriété (*CODEPP*) autant de fois qu'il y a de propriétés dans l'enregistrement.

Enfin, la description de l'enregistrement se termine par le mot-clé *fmajeg*.

REMARQUES :

(1) Un enregistrement regroupe au moins une propriété.

(2) Lors de la déclaration d'un enregistrement par *crefi*, *majfi* ou *majegdo* les caractéristiques FQUOI (fichier concerné) et DQUOI (document concerné) de l'enregistrement sont mises à jour automatiquement.

(3) Les enregistrements ayant leurs listes FQUOI et DQUOI vides seront supprimés automatiquement car ils ne seraient alors rattachés à aucun fichier ou document.

3.f. Remarque sur l'utilisation du modèle lors de l'analyse organique

Résumons-nous ; au niveau de la réalisation logique d'une application, trois situations sont possibles :

- L'application est réalisée avec des moyens classiques de programmation (PL/1, Cobol, ...) : alors la spécification logique des opérateurs sera faite sous forme de schémas de programmes en Socrate et l'implémentation sera réalisée sous forme de programmes en PL/1 ou Cobol avec des définitions des fichiers.

- L'application utilise une base de données Socrate : alors il y aura une première base pour exprimer la logique des opérateurs, ainsi que la définition éventuelle d'une deuxième base prenant en compte les problèmes de performance : taille de pages, collision sur l'espace réel, multiplicité des accès, ...

- On peut aussi avoir une application mixte qui utilise une base de données et au niveau des entrées-sorties, avoir des programmes classiques ayant des fichiers de manoeuvre comme interface avec la base.

- Il est aussi possible de définir un prototype (cf. p. 2.2) qui sera décrit en Socrate.

Afin de mieux décrire les options que nous avons prises
. pour la phase fonctionnelle (chapitre 1),
. pour la phase organique que nous venons d'introduire,
. et pour le passage d'une phase à l'autre (schéma du chapitre 2 p. 2.2),

proposons le tableau suivant.

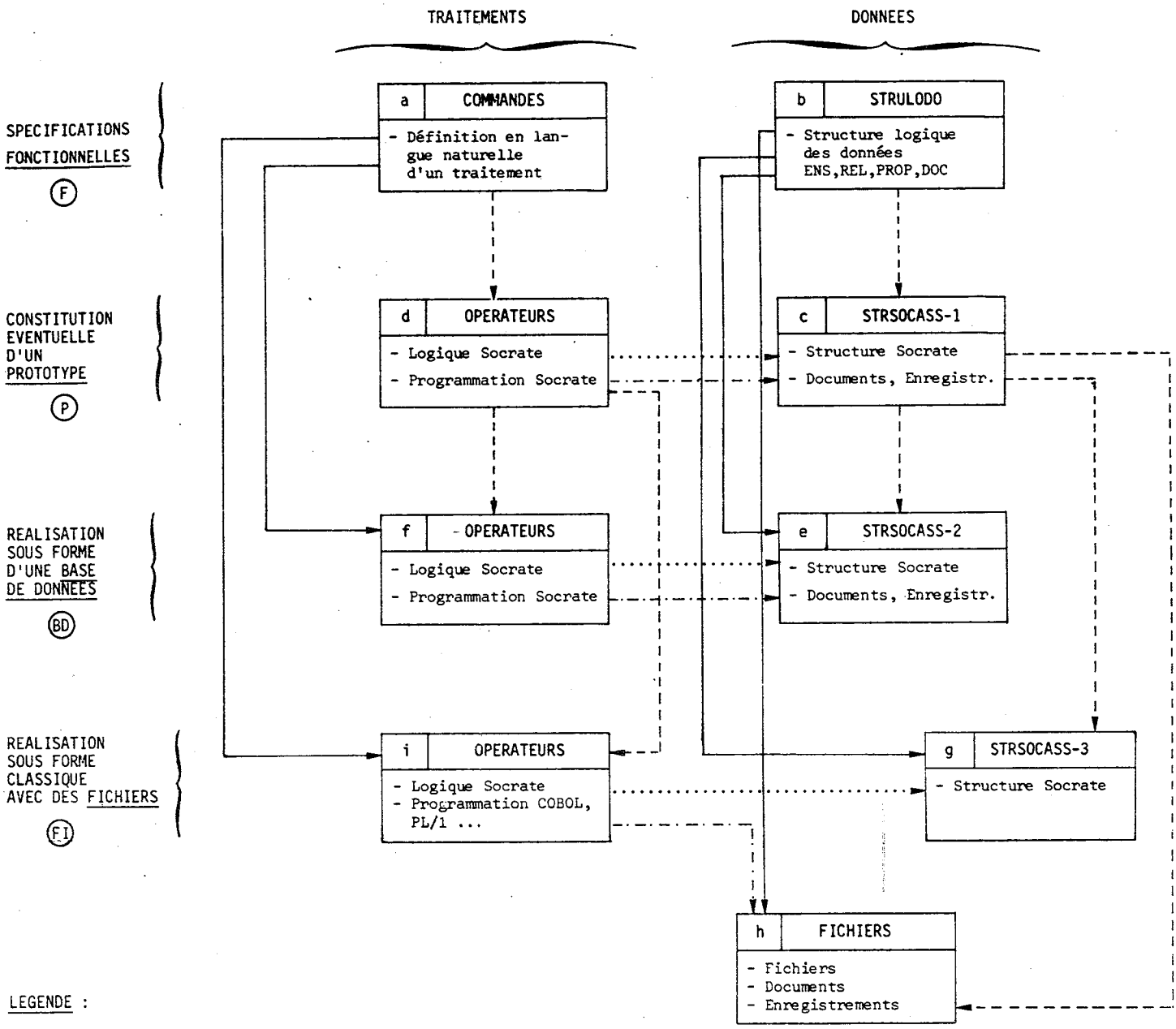

TRAITEMENTS
DONNEES
SPECIFICATIONS FONCTIONNELLES
(F)
CONSTITUTION EVENTUELLE D'UN PROTOTYPE
(P)
REALISATION SOUS FORME D'UNE BASE DE DONNEES
(BD)
REALISATION SOUS FORME CLASSIQUE AVEC DES FICHIERS
(FI)
a COMMANDES
- Définition en langue naturelle d'un traitement
b STRULODO
- Structure logique des données ENS,REL,PROP,DOC
d OPERATEURS
- Logique Socrate
- Programmation Socrate
c STRSOCASS-1
- Structure Socrate
- Documents, Enregistr.
f OPERATEURS
- Logique Socrate
- Programmation Socrate
e STRSOCASS-2
- Structure Socrate
- Documents, Enregistr.
i OPERATEURS
- Logique Socrate
- Programmation COBOL, PL/1 ...
g STRSOCASS-3
- Structure Socrate
h FICHIERS
- Fichiers
- Documents
- Enregistrements
LEGENDE :
Déduction : implémentation des commandes sous forme d'opérateurs ou de la structure logique des données.
Déduction supplémentaire dans le cas où un prototype est construit.
Relations entre traitements et données pour exprimer la logique des opérateurs.
Relations entre traitements et données pour exprimer la programmation de ces opérateurs.

FIGURE 3-1

COMMENTAIRE :

Le système MACSI-1 peut supporter plusieurs formes de déroulement d'une application.
Les utilisateurs ont la possibilité de construire un "prototype" (P) ; ils ont le choix entre une réalisation de leur application sous forme d'une base de données (BD) ou avec des fichiers (FI) ou avec une combinaison de ces deux formes.
Ces possibilités diverses peuvent s'exprimer simplement par l'expression logique suivante :

une application sous MACSI-1 ≡ F et (P ou rien) et ((BD ou FI) ou (BD et FI))

≡ F ; [P] ; { BD / FI / BD,FI }

CHAPITRE 4

IMPLEMENTATION DES PROGRAMMES DE GESTION DES SPECIFICATIONS DE MACSI-1

4.a. Choix fondamentaux et réalisations

Le formalisme de description de traitements dans MACSI-1 a été décrit dans le chapitre 3. Or, l'outil MACSI-1 est lui-même un produit informatique qui doit donc pouvoir être implémenté en retenant les mêmes objectifs que ceux exposés au chapitre 3, à savoir :

- une programmation structurée et descendante des programmes ;
- des programmes qui expriment clairement la logique des traitements ;
- des programmes faciles à écrire et à modifier ;
- des informations structurées sous forme de base de données.

Ces normes de réalisations ont été adoptées pour pouvoir satisfaire certains objectifs de MACSI-1 : MACSI-1 doit proposer un outil adaptable à l'analyse de différentes applications, en tenant compte des multiples conditions et contraintes qui font de chaque projet un cas particulier. Plutôt que de présenter une "méthode d'analyse assistée par ordinateur", comme celles qui existent sur le marché, nous avons voulu réaliser une méthode qui aurait la qualité de pouvoir s'adapter au problème posé par chaque projet particulier. Pour cela, il faut que des extensions, des modifications ou des suppressions de composants du modèle ou de règles de description, ainsi que des programmes d'aide à l'analyse correspondants soient possibles sans un gros effort de programmation. Il fallait ainsi éviter la critique la plus fréquente faite aux "packages" quant à leur trop grande rigidité dans leur conception et leur réalisation.

Par opposition, on peut dire que le projet MACSI-1, si ces objectifs étaient totalement atteints, se veut "une méthode de construction de méthodes d'analyse" [22]. Le module MODMACSI (modification de MACSI-1) évoqué au chapitre 2 (p. 2.2) témoigne de ce souci prioritaire de flexibilité. Si la méthode de description est jugée par un chef de projet comme inadéquate pour le projet dont il a la charge, il doit pouvoir définir lui-mêmes certaines modifications du modèle de représentation du système d'information, puis implémenter facilement les modifications correspondantes des règles de description et des programmes d'aide à l'analyse.

Pour obtenir cette flexibilité, nous avons donc pris certaines options techniques pour la programmation et l'implémentation de MACSI-1, dont voici une énumération sommaire :

1) La mise en oeuvre d'un système de gestion de base de données (§ 4.b.1.) : les raisons de ce choix ont été largement évoquées au chapitre 3.

2) Une gestion des codes pour constituer une nomenclature des éléments du projet (§ 4.b.2.).

3) Un système de protection des spécifications et des zones de travail privées pour chaque utilisateur "on line", permettant à plusieurs analystes de travailler simultanément sur la base de MACSI-1 (§ 4.b.3. et § 4.b.4.).

4) Une gestion souple des textes et des messages grâce à un éditeur de textes (§ 4.b.5.).

5) Une définition d'un certain nombre de programmes de services généraux pour diminuer, autant que faire se peut, la longueur et la complexité d'écriture des programmes de MACSI-1 (§4.d.).

6) Considérant la structure d'automate d'état fini déterministe sous-jacente à toutes les règles de description, réalisation d'un analyseur syntaxique paramétré (§ 4.e.).

Avant d'examiner en détail ces différents aspects d'implémentation de MACSI-1, précisons tout d'abord que le système de gestion de bases de données Socrate a été choisi pour réaliser le système MACSI-1. Les raisons de ce choix ont été exposées au chapitre 3 lorsque nous avons proposé le système Socrate pour représenter la structure logique des données d'une application quelconque et pour écrire les opérateurs associés. Le système MACSI-1 se compose donc d'une structure Socrate (§ 4.b.1.) et d'un ensemble de programmes écrits en Socrate (ainsi que quelques programmes Cobol) (§ 4.c.).

Il existe à Grenoble plusieurs versions de Socrate :

- à l'IMAG
  - . une version prototype de Socrate développée par l'équipe de J.R. ABRIAL sur IBM-360/CP-CMS (cf. [30]),
  - . une version batch IBM-360 diffusée par ECA-Automation (cf. [27]),
  - . une version IRIS 80/SIRIS 7-8 (cf. [28]).
- à l'I.U.T.-B
  - . une version IRIS 45-50/SIRIS 2-3 (cf. [29]).

Nous avons choisi cette dernière version sur IRIS 45-50 pour obtenir une certaine diffusion de MACSI-1 autorisée par la portabilité de cette version de Socrate.

Ce système Socrate SIRIS 2-3 peut gérer, à partir d'un même calculateur, plusieurs bases contenant chacune plusieurs usagers, grâce à un fichier (au sens Socrate) appelé 'Base des bases" (BDB).

Cette BDB contient, entre autres informations, les noms des différentes bases qui en font partie, ainsi que les noms des utilisateurs de chaque base et leurs droits d'accès. Ce qui est important de souligner ici c'est que pour une application donnée (une base Socrate) plusieurs utilisateurs peuvent y accéder simultanément, c'est-à-dire un même projet peut être spécifié par plusieurs personnes à partir de terminaux différents.

4.b. La structure de données

4.b.1. La structure Socrate de MACSI-1

Nous avons vu au Chapitre 1 quels étaient les composants du modèle du système d'informations retenu pour spécifier une grande famille d'applications informatiques. Ces composants sont implémentés pour constituer une structure Socrate associée à MACSI-1. Le lecteur trouvera une copie de cette structure en ANNEXE A à laquelle il pourra faire référence.

Nous avons défini cette structure d'après les principes suivants :

- les composants du modèle sont des entités de plus haut niveau de la structure
- les propriétés de ces composants sont des caractéristiques de ces entités.

Ce mécanisme a déjà été présenté en § 3.a.. Le processus de déduction et les règles employées pour établir la structure Socrate de données de MACSI-1 ont été en général les mêmes que celles proposées aux utilisateurs de MACSI-1 pour déterminer la structure Socrate associée à la structure logique de données de leur application informatique.

Le lecteur remarquera dans les entités de plus haut niveau de la structure Socrate de MACSI-1 qu'une caractéristique appelée PROJET apparaît systématiquement. Elle permet l'enregistrement des spécifications de plusieurs projets dans une même base de données.

Par exemple, nous avons utilisé la caractéristique PROJET = 1 pour distinguer dans la même base les spécifications de MACSI-1. En effet, bien que le modèle de système d'information nous permette de spécifier un bon nombre d'applications en informatique de gestion, on peut distinguer la même structure dans tout projet informatique :

* Des services administratifs = - équipe d'étude informatique
  - le client
  - le constructeur
  - l'équipe d'exploitation.

* Des personnes qui participent aux projets.

* Des actions = - des interviews
  - des programmes à réaliser
  - des constructions de jeux d'essai.

* Des données = le contenu du dossier d'analyse.

* Des programmes de gestion du dossier d'analyse.

* etc ...

Nous avons donc utilisé les composants du modèle pour spécifier un projet informatique particulier : la propre implémentation de MACSI-1.

PROJET = 2, 3 ... nous permet d'identifier plusieurs projets ou applications stockés sur la même base.

Cette caractéristique PROJET nous permettra de contrôler les droits d'accès de différentes personnes au système MACSI-1 ou aux différentes applications spécifiées par MACSI-1 (cf. § 4.d.3.2.).

4.b.2. Gestion des nomenclatures

Au cours des chapitres précédents et notamment au § 1.c.2., nous avons souligné l'importance des l'unicité d'un code d'identification pour tous les éléments de tous les constituants d'un projet. Ce code permet une différentiation non seulement des éléments d'un même composant mais aussi de tous les éléments de tous les composants du sujet. Le premier cas pourrait être résolu par Socrate lui-même par l'utilisation de la nature discriminante d'une caractéristique mais pour s'assurer que, dans le second cas, tout code est bien discriminant et pour constituer une nomenclature des éléments du projet, un nouveau composant de MACSI-1, le DICTIONNAIRE, a été créé (l'entité DICO pour DICtiOnnaire). Dans le Dictionnaire, nous stockons tous les codes d'identification des éléments du projet. C'est la caractéristique CODE qui permet d'enregistrer la valeur du code d'identification des éléments. Il est bien évident qu'il doit être de type discriminant au sens Socrate. Le Dictionnaire nous permet aussi d'enregistrer la personne qui a déclaré ou défini un élément. Ceci est intéressant car une mise à jour ou une modification éventuelle d'un élément ne peut se faire que par la même personne qui l'a déclarée ou définie. Cette personne est enregistrée dans la caractéristique INITIALISATEUR. La caractéristique NATURE nous permet de répertorier le type d'élément auquel correspond le code. Par exemple : nature = 1 indique le code d'un service administratif ; nature = 7 est un document, etc ....

Le fait de dire qu'on peut associer plusieurs textes explicatifs à un même élément (cf. § 1.d.5.) justifie l'existence de l'entité LISTE-EXPLIC. Elle nous permet de répertorier la liste des explications associées à l'élément aussi bien que l'ordre d'importance des explications. En effet, on peut très bien envisager, par exemple, une explication ou définition résumée de l'élément qui serait la première à consulter. Si celle-ci ne donne pas satisfaction à l'utilisateur ou bien s'il désire

des explications supplémentaires, il pourra demander une deuxième explication plus complète.

4.b.3. Système de protection des spécifications

Plusieurs utilisateurs peuvent accéder simultanément à la base de données de MACSI-1 ; aussi, nous avons dû concevoir un système de protection des données. En effet, les composants d'un projet informatique (les entités Personnes, Actions, Commandes, etc ...) "passent" par des états différents tout au long de l'analyse. Un composant qui est déclaré sera plus tard complètement défini (par la commande suivante, ou le lendemain, ou dans 3 mois ... !). Il faut que ce soit la même personne qui fasse ces deux opérations, mais il faut aussi empêcher que d'autres personnes déclarent ou citent des composants ayant le même code que celui-ci. C'est par la caractéristique TOPVAL en combinaison avec l'entité DICO que nous avons implémenté un système de protection ou verrouillage sur les éléments des composants (réalisation des entités). En effet, toutes les entités, fonctionnelles ou de travail, qui peuvent être accédées par plusieurs utilisateurs ont une caractéristique appelée TOPVAL. Le TOPVAL nous définit l'état de développement d'une réalisation d'entité et par conséquence les droits d'accès à cette réalisation. Les différentes valeurs que prend le TOPVAL d'une réalisation d'entité tout au long de son existence dans le système sont les suivantes :

1 = En cours de déclaration
2 = Déclarée
3 = En cours de mise à jour
4 = Erreur reconnue par macro de bilan
5 = En cours de modification
6 = Etat privé
8 = Etat public.

Voyons plus en détail la signification de chacune de ces valeurs.

Les TOPVAL impairs représentent un blocage complet de la réalisation de l'entité. Ils indiquent qu'un travail est en cours sur la réalisation de l'entité. Les TOPVAL pairs représentent des états stables

de la réalisation.

TOPVAL = 1. Lors de la création d'un élément ou bien de la déclaration de celui-ci par la création ou mise à jour d'un autre élément, il se produit des mises à jour en rapport avec le TOPVAL :

a) La saisie du code de l'élément commande la génération d'une réalisation du DICO pour ce code (CODE du DICO = code de l'élément).

b) Une réalisation de l'entité correspondant à la nature de l'élément est créée avec la mise à jour correspondant au code de l'élément. Le TOPVAL sera égal à 1.

TOPVAL = 2. La déclaration de l'élément se termine par la spécification de son libellé. Alors les mises à jour dans la base sont les suivantes :

a) Mise à jour de l'INITIALISATEUR de l'élément dans le DICO associé.

b) Mise à 2 du TOPVAL de l'élément.

Une réalisation d'entité ayant son TOPVAL = 2 signifie qu'elle a été déclarée. Elle est prête pour sa mise à jour ou sa définition complète ainsi que pour un attachement à d'autres éléments. Par exemple, si l'action A20 a été déclarée (code et libellé spécifiés, TOPVAL = 2) par la mise à jour du module M5, on a le droit de citer cette action comme étant l'un de ses éléments dans la mise à jour d'un autre module M10. Les liaisons correspondantes seront établies.

Mais il faut préciser que c'est la même personne qui a déclaré l'élément qui sera autorisée à faire une opération de mise à jour ou modification de celui-ci.

TOPVAL = 3. Au début de la mise à jour d'un élément par les commandes correspondantes et par la personne qui l'a déclarée, le TOPVAL est mis à 3. A la fin de la mise à jour, il peut devenir égal à 6 ou égal à 8.

TOPVAL = 6. Correspond à la mise en accès privé de la réalisation de l'entité. En effet, à la sortie d'une création, mise à jour ou modification d'un élément, on a la possibilité de spécifier si l'élément est en accès privé ou public. L'accès privé interdit l'utilisation de l'élément, soit pour le modifier, soit pour s'y attacher par une personne autre que l'initialisateur. C'est généralement le cas prévu par l'initialisateur lorsque les spécifications qu'il vient de donner ne sont pas définitives.

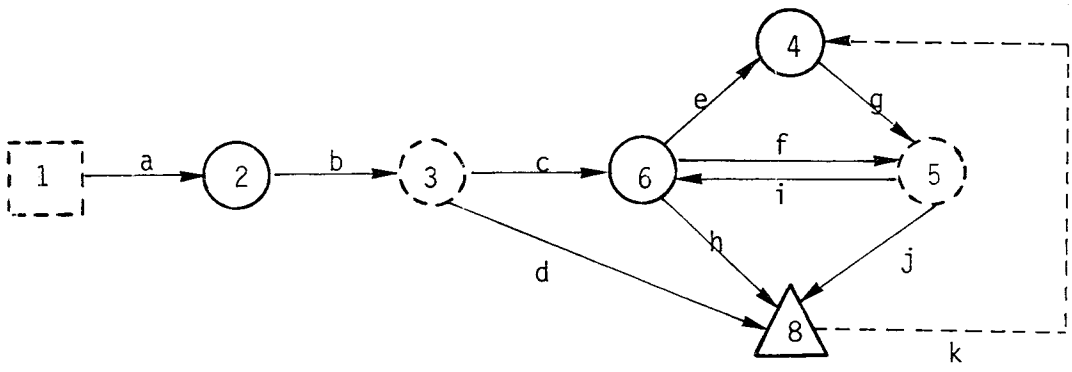
TOPVAL = 8. L'initialisateur autorise l'accès et le rattachement de cet élément à tous les utilisateurs de la base. C'est l'accès public. Il déclare que les spécifications sont définitives, donc aucune modification n'est plus possible. Une publication des spécifications de l'élément accompagne le changement en état 8 de façon à ce que les autres analystes puissent prendre connaissance des spécifications de l'élément et l'utiliser éventuellement dans leurs propres spécifications.

L'administrateur de la base, généralement le chef du projet ou le responsable informatique est le seul qui pourra éventuellement modifier une réalisation d'entité qui est en accès public, car il est le seul à pouvoir utiliser le langage de requêtes de Socrate pour interroger et modifier la base de données : il peut alors savoir quels sont les analystes qui se servent d'un élément public en particulier et faire une enquête auprès d'eux pour prendre les mesures nécessaires.

TOPVAL = 4. Cette valeur indique qu'une erreur a été détectée par une macro-instructions de bilan. Par exemple, une commande d'édition qui n'a pas un document associé, un groupe d'exécution qui n'a pas de membres, etc ... verront leurs TOPVAL basculer à l'état 4. Evidemment, ce passage à l'état 4 n'est possible que pour les réalisations d'entités qui étaient déjà à l'état 6.

TOPVAL = 5. Au début de la modification d'un élément par son initialisateur, le TOPVAL prend la valeur 5. A la fin de la modification, l'initialisateur peut établir l'accès sur l'élément qui sera défini public ou privé.

Nous pouvons illustrer les changements d'états des éléments d'un projet par un graphe de transition :



Etat stable

Etat transitoire

Etat définitif

Transitions dirigées par les commandes utilisateurs

Transition effectuée par l'administrateur de la base

Ces transitions (les arcs a, b, c, ...) seront gérées par des programmes du système MACSI-1 décrits en § 4.c. :

Exemple : L'arc "a" est géré par le programme INIT.

4.b.4. Zone de travail des opérateurs de MACSI-1 = Le DICINIT

Le DICINIT est une zone de travail de nature purement organique utilisée uniquement par les opérateurs de MACSI-1 notamment pour saisir et sauvegarder des valeurs, mais aussi, pour effectuer certains contrôles sur les utilisateurs de MACSI-1.

Etant donné la possibilité d'accès de plusieurs utilisateurs simultanément sur la même base, nous assignons à chacun une réalisation de l'entité DICINIT.

Résumons les différentes fonctions du DICINIT :

a) Contrôle : des contrôles seront possibles en stockant dans le DICINIT :

- le code de la personne qui entre et qui travaille dans le système MACSI-1
- le code du projet sur lequel elle travaille (PROJET)
- la date du jour (DATE)
- le code de l'élément sur lequel elle travaille (CODE-EN-COURS).

b) Saisie : le DICINIT contient des caractéristiques portant des noms significatifs se rapprochant autant que possible des mots-clés utilisés dans les commandes de création, mise à jour, modification et suppression définies dans les chapitres précédents. Ces caractéristiques seront utilisées par exemple pour stocker temporairement l'information saisie en mode conversationnel (cf. macro SAISIE, § 4.d.1.).

c) Stockage d'adresse : des pointeurs sur les entités de la structure permettront le stockage des adresses. En effet, le nombre limité de variables de type adresse "Xi" en Socrate est un problème pour la programmation des opérateurs de MACSI-1. Nous avons alors été amenés, dans certains cas, à transférer leurs valeurs sur des caractéristiques du DICINIT pour les libérer.

d) Stockage de sous-ensembles : des chaînes d'inverses sur les entités de la structure permettent en particulier de faire des saisies multiples. En général, on peut travailler complètement au niveau de la chaîne d'inverse du DICINIT, c'est-à-dire stocker les adresses des éléments déclarés par une création, mise à jour ou modification, et puis réaliser les liaisons si les opérations sont validées après demande de confirmation (mode conversationnel).

e) Sauvegarde : l'entité CODCREE permet de stocker les codes des éléments déclarés par une manipulation. Ceci permet de lister à l'utilisateur les codes des éléments déclarés au cours d'une opération. L'entité CODE-A-LISTER permet la sauvegarde des codes des éléments, qui ont été mis à jour dans une période de temps. Cette zone est utilisée par l'administrateur de la base pour lister les éléments définis depuis la dernière édition afin de maintenir sa documentation à jour.

f) Zone de travail : l'entité SAVEX du DICINIT sera utilisée comme zone de travail de l'éditeur de textes de MACSI-1 (cf. § 4.b.5.).

4.b.5. Gestion des textes

Une grande partie des spécifications se fait sous forme de texte libre en langue naturelle. C'est le cas notamment des définitions que l'on donne à chaque élément du projet, mais aussi les textes des conditions d'exécution des éléments d'un module, le résultat d'une action, les modifications des commandes ou des opérateurs, etc.... Toutes ces caractéristiques de type TEXTE ont été regroupées par commodité de travail et pour des raisons de performances dans une zone spéciale appelée EXPLICATION (entité EXPLIC). C'est sur ces Explications que nous avons défini l'éditeur de textes de MACSI-1.

Chaque texte ou Explication a un code d'identification numérique (CODEX). Ce code ne sera pas donné par l'auteur du texte, mais il sera géré par MACSI-1 et communiqué à l'auteur.

La caractéristique AUTEUR nous permet de conserver l'identification de la personne qui a spécifié le texte car elle sera la seule à pouvoir éventuellement utiliser l'éditeur sur ce texte.

Les textes ou explications qui correspondent à la définition d'un élément du projet ne peuvent pas être supprimés sauf si l'élément est supprimé lui-même. Cette restriction est définie par la caractéristique SUPPRESSION.

Toute explication peut être explicative d'un ou plusieurs éléments (par exemple, le texte définissant la FONCTION d'une personne dans un groupe d'exécution précise à la fois la PERSONNE et le GROUPE D'EXECUTION concerné). La liste des éléments concernés par l'explication est répertoriée dans LISTE-DICO. Ainsi, pour faciliter au maximum la recherche documentaire sur les textes, les relations entre tout élément du projet répertorié dans DICO et les textes qui lui sont associés dans EXPLICATION sont enregistrés dans les deux sens. Finalement, le CONTENU d'un texte ou explication est représenté par une suite de 100 lignes au plus et de 60 caractères maximum par ligne.

Voyons maintenant les caractéristiques de l'éditeur de textes.

Le système Socrate que nous utilisons possède son propre éditeur de textes (cf. [29]) qui permet à l'utilisateur la possibilité de créer, lister, mettre à jour les textes précités par insertion ou effacement de lignes et changement de chaînes de caractères sur une ligne. Cet éditeur de textes est placé dans le système Socrate au même niveau que le macrogénérateur, le processeur de requêtes, etc ...

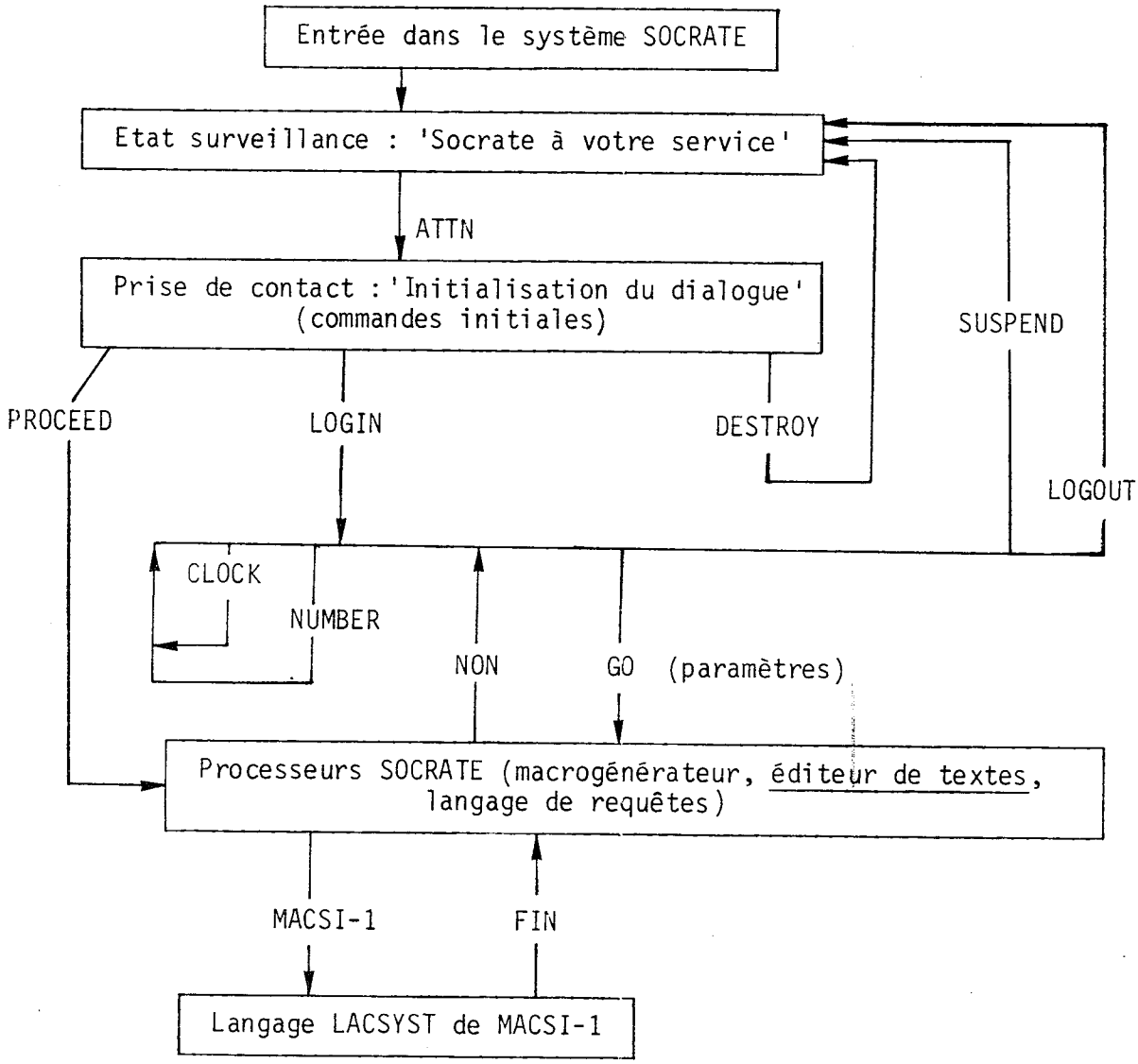
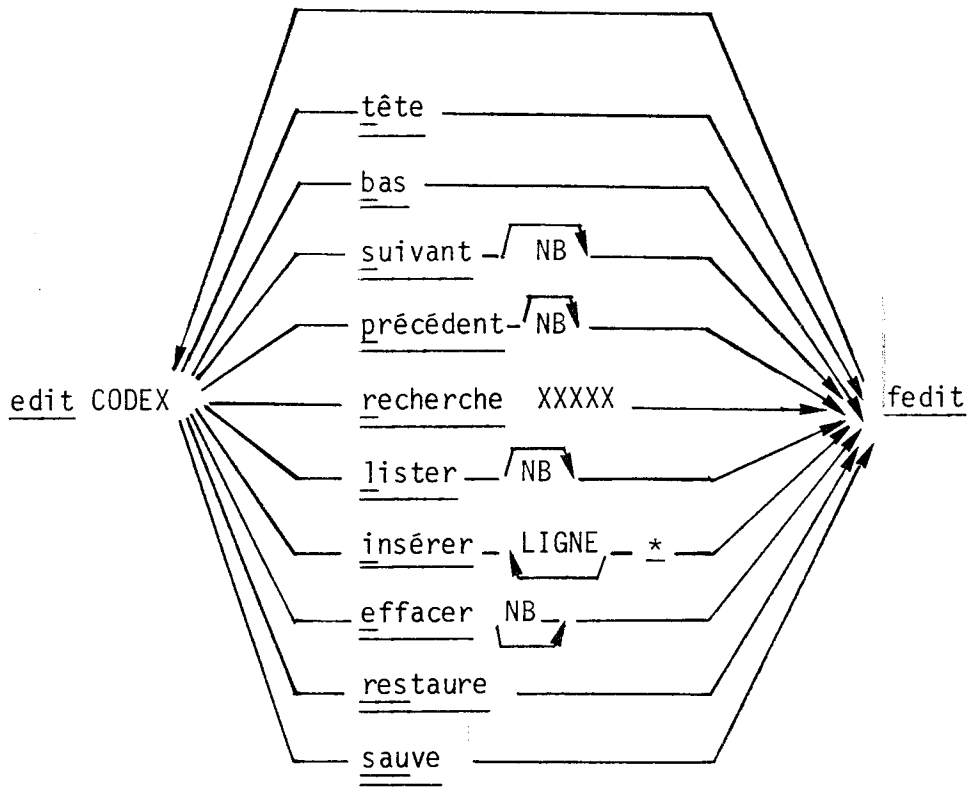


Figure 4-1

En effet, nous voyons dans la figure 4-1 les différents niveaux du système Socrate. Notons qu'à partir du niveau des processeurs Socrate le mot-clé MACSI-1 nous permet d'accéder au langage de spécifications LACSYST. C'est dans cet environnement que vont se dérouler toutes les opérations de spécification sous MACSI-1. L'éditeur de texte Socrate n'est pas à la disposition de l'utilisateur car il faudrait alors gérer les allers et retours entre MACSI-1 et les processeurs Socrate.
Seul un informaticien connaissant le système Socrate peut faire ces manipulations. Or, il ne faut pas oublier que notre produit est destiné en priorité à des utilisateurs non informaticiens. Pour cette raison, nous avons défini un éditeur de textes sous MACSI-1. Cet éditeur permettra d'accéder et de modifier les textes du projet enregistrés dans l'entité Explication.

Nous pouvons définir les fonctions de cet éditeur par le graphique suivant :



REMARQUES :

Le texte à éditer est recopié dans une zone de travail appelée SAVEX. Un pointeur permet d'identifier la "ligne courante" du texte. Au moment de la recopie du texte à éditer,une ligne vide (LV1) de début du texte et une ligne vide (LV2) de fin du texte sont concaténées aux extrémités du texte. Le pointeur de ligne courante est alors placé sur la première ligne réelle au début du texte (sauf si le texte initial est vide, auquel cas le pointeur est positionné sur la ligne vide LV1). Le texte initial reste inchangé pendant les manipulations qui se font sur cette zone de travail. C'est l'utilisateur qui décidera du remplacement éventuel du texte initial par le texte transformé dans la zone de travail.

FONCTIONS DE L'EDITEUR :

a) Instructions de déplacement du pointeur indicateur de la ligne courante

tête = place le pointeur au début du texte avant la première ligne réelle, c'est-à-dire que le pointeur est positionné sur la ligne vide de début du texte LV1. Le message EN TETE est édité par le système.

bas = place le pointeur à la fin du texte, c'est-à-dire sur la dernière ligne réelle du texte. Cette ligne sera éditée par le système.

suivant [<NB>] = permet de déplacer le pointeur vers le bas du texte du nombre de lignes indiquées par le chiffre NB. Ce nombre est optionnel (égal à 1 par défaut) . La ligne courante après le déplacement sera éditée. Si le nombre de lignes fait que le déplacement dépasse la dernière ligne du texte, le message EN BAS est édité et le pointeur reste positionné sur la ligne vide LV2.

précédent [<NB>] = permet de remonter le pointeur du nombre de lignes indiqué par le chiffre NB. Si ce nombre est omis, la valeur par défaut est égale à 1. Si le déplacement dépasse la première ligne réelle du texte, le message EN TETE est imprimé et le pointeur est placé sur la ligne vide LV1.

recherche XXXXX = cette instruction permet la recherche à partir de la ligne courante, de la première ligne du texte commençant par les 5 premiers caractères indiqués par XXXXX (chaîne de 5 caractères). Si l'éditeur rencontre cette ligne, elle sera éditée et le pointeur placé dans cette position. Si la recherche n'aboutit pas avant la fin du texte,

le message EN BAS sera édité et le pointeur sera positionné sur la première ligne réelle du texte.

b) Instruction de listage du texte

lister [<NB>] = permet l'impression du nombre de lignes indiquées par NB, ligne courante comprise. Si NB qui est optionnel est omis, seule la ligne courante sera imprimée. Si le nombre de lignes indiqué par NB excède le texte, après l'édition du texte le message EN BAS est imprimé et le pointeur se trouvera positionné sur LV2.

c) Instructions de modification du texte

insérer LIGNE * = cette instruction permet l'insertion d'une ou plusieurs lignes entre la ligne courante et la ligne suivante. L'astérisque (*) comme dernier caractère d'une ligne indique la fin de l'opération d'insertion. Chaque ligne insérée sera imprimée. Après une insertion de lignes, le pointeur sera positionné sur la dernière ligne insérée.

effacer [<NB>] = permet l'effacement du nombre NB de lignes, ligne courante comprise. Le nombre de lignes est optionnel et la valeur par défaut est égale à 1, c'est-à-dire que seule la ligne courante sera supprimée si le nombre de lignes NB est supérieur au nombre de lignes du texte à partir de la ligne courante, toutes ces lignes seront effacées, le message EN BAS sera édité et le pointeur sera placé sur la ligne vide LV2.

d) Instructions de sauvegarde ou annulation des modifications

restaure = permet l'annulation de toutes les manipulations qui ont été faites depuis la dernière utilisation de l'instruction sauve (voir § suivant) ou de l'entrée dans l'éditeur si aucun sauve n'a été fait. Après cette instruction restaure, nous revenons au niveau des commandes de l'éditeur.

sauve = permet la sauvegarde des transformations faites par l'éditeur. Après l'utilisation de sauve, on reste dans l'éditeur, c'est-à-dire que toutes les fonctions de l'éditeur sont disponibles.

fedit = permet la sortie de l'éditeur en sauvegardant les transformations faites. Le nouveau texte remplace l'ancien qui n'est plus disponible.

NB : Il existe, dans notre structure, une autre entité qui regroupe des textes, c'est l'entité MESSAGE. Elle est purement organique et nous nous en servons pour stocker tous les messages qui sont édités par les opérateurs ou programmes de MACSI-1. Deux avantages sont à souligner :

- ce regroupement des messages nous donne une meilleure projection dans l'espace physique,

- si le contenu d'un message édité par un opérateur doit être l'objet d'une modification, il suffira de le modifier dans l'entité MESSAGE sans par la suite recompiler l'opérateur associé.

4.c. Deux modes de fonctionnement des programmes de MACSI-1

Nous allons présenter aux lecteurs les choix techniques et les méthodes employées dans l'implémentation des programmes de MACSI-1. Mais auparavant, il faut souligner que le système MACSI-1 opère selon deux modes de travail :

- Mode "conversationnel"
- Mode "batch".

En effet, une version conversationnelle de MACSI-1 est nécessaire pour assurer la formation des analystes à l'utilisation de la méthode. A toute méthode d'analyse est associée une pédagogie conçue pour apprendre aux utilisateurs qui auront à se servir de la méthode, la définition des concepts de base et l'utilisation efficace des moyens qu'elle offre. On constate parfois que les efforts intellectuels et financiers employés pour la conception et la mise en pratique d'une telle pédagogie sont du même ordre de grandeur que les efforts de conception et d'implémentation de la méthode elle-même.

Le mode conversationnel de travail doit donc guider l'utilisateur dans les spécifications en lui permettant une consultation aisée des définitions des différents concepts de base. Une utilisation répétée du système en mode conversationnel donne rapidement à l'utilisateur les éléments nécessaires pour continuer les spécifications en mode batch.

Mais ce mode conversationnel a un autre intérêt. En effet, certaines opérations complexes, notamment les modifications et les suppressions, se font en mode conversationnel. La quantité des modifications et mises à jour dans la base que ces opérations impliquent, demande une récapitulation, de la part du système, de ces opérations afin que l'utilisateur puisse les analyser et ensuite les valider.

Dans l'optique de formation des utilisateurs de MACSI-1, deux options différentes d'utilisation du conversationnel ont été développées :

- Une option "bavarde" qui donne de nombreuses explications à l'utilisateur avant et après une saisie.

- Une option "silencieuse" qui se contente de demander la saisie des informations en utilisant les mots-clés correspondants (TYPE, CODE, ACCES, ...).

Il faut souligner que l'utilisateur a la possibilité à tout moment (quand il a "la main") de passer d'un mode à l'autre, c'est-à-dire que s'il est en option bavarde, il peut, en tapant un caractère spécial, passer en option silencieuse et réciproquement. Ceci est très important car l'option bavarde est très pédagogique mais elle devient rapidement très lourde à utiliser lorsque l'utilisateur a pris une certaine pratique. Il pourra repasser en option bavarde dès qu'il le voudra pour avoir plus de précision sur une certaine opération.

Le lecteur trouvera en ANNEXE C un exemple de déroulement d'une session en conversationnel où les deux options, "bavarde" et "silencieuse", sont combinées.

Par contre, nous pensons que les trois chapitres précédents ont été suffisamment significatifs pour montrer au lecteur l'importance et la quantité des spécifications d'un projet. Il devient alors très difficile d'effectuer un tel travail en conversationnel. Les analystes auront la possibilité de donner leurs spécifications en mode batch ou par lots, c'est-à-dire sur une feuille de papier ils décriront, en format libre mais suivant une syntaxe bien précise (définie par les Règles de Description), les opérations souhaitées et les données nécessaires. Les chapitres 2 et 3 sont des exemples de cette utilisation.

4.d. Méthode de programmation

4.d.1. Description de quelques programmes de service

Ces programmes de service ont été conçus comme des utilitaires pour les programmes de MACSI-1. Ils ont tous une fonction bien définie et connue par les différents membres de l'équipe qui ont implémenté MACSI-1 de façon à faciliter, homogénéiser et structurer la programmation.

Nous décrirons quelques utilitaires employés pour la programmation du mode conversationnel, mais il est bien évident que des programmes de service équivalents, assurant les mêmes fonctions, ont été utilisés pour la programmation du mode batch (cf. 4.e.).

a - L'utilitaire de SAISIE

Objet : - Saisir une caractéristique explicite à la console et la restituer dans une variable de type Z.
- Edition d'un message si on travaille en option bavarde.
- Gestion du changement d'option (bavarde - silencieuse).

Syntaxe :

Saisie NOM-CAR Zi NN

REMARQUES :

(1) NN = indique le numéro du message explicatif de la saisie (voir MESSAGE en § 4.b.5.) à éditer en option bavarde.
Si NN = 0, il n'y a pas édition de message.

(2) NOM-CAR = identificateur à afficher à la console avant de faire la saisie
Il doit figurer dans le DICINIT.

(3) Z_i = Variable de type Z qui sera mise à jour avec le contenu de la saisie.

Schéma de programme :

```
          defmac SAISIE &1 &2 &3  /* &i = paramètre */
               faire
(1)                 COMMENTAIRE &3

(2)                 m &1 de un DICINIT Y22 = EXT


(3)                 m &2 = &1 de un DICINIT Y22
(4)                 si &2 = '/' alors M Y21 = -Y21
                                      refaire
                    fin
               fin
          fdef
```

COMMENTAIRES :

(1) Edition du message n° &3 si le système est en option bavarde.

(2) Saisie dans &1 (caractéristique du DICINIT).
Y22 = numéro du DICINIT de l'utilisateur.

(3) Transfert de la valeur de &1 dans &2 (Z_i)

(4) Changement d'option (bavarde - silencieuse) si l'utilisateur a tapé le symbole '/'

Exemple d'utilisation :

Supposons qu'il existe dans la base :

- un message 17 dont le libellé est :
  FRAPPEZ LA DATE SOUS FORME JJMMAA
- un message d'erreur 21 dont le libellé est :
  JOUR INCORRECT
  et dont l'explication associée est :
  UN JOUR DOIT ETRE COMPRIS ENTRE 1 ET 31.

Pour la séquence suivante :

```
faire
   SAISIE DATE Z1 17
                  m Y1 = Z1
   si y1 < 1 ou y1 > 31
      alors ERREUR 21 refaire
   fin
fin
```

on obtiendrait à la console :

Option silencieuse	Option bavarde
DATE : 410375	FRAPPEZ LA DATE SOUS FORME JJMMAA
JOUR INCORRECT	DATE : 410375
DATE : 310375	JOUR INCORRECT
	EXPLICATION
	OUI
	UN JOUR DOIT ETRE COMPRIS ENTRE 1 ET 31
	DATE : 310375

b - COMMENTAIRE

Objet : Edition d'un message en option bavarde ; en option silencieuse, COMMENTAIRE équivaut à une instruction nulle.

Syntaxe : COMMENTAIRE NN

NN = numéro du code du message à éditer. Si NN = 0, COMMENTAIRE équivaut à une instruction nulle (même en option bavarde).

c - NORMCOD

Objet : Normalisation des codes par élimination des blancs, recadrage à gauche et complètement à droite par indéfini.

Syntaxe :

normcod Z_i

Remarques :

(1) Z_i = Variable de type Z contenant en entrée la donnée à normaliser et en sortie la donnée normalisée.

(2) Généralement, on utilise NORMCOD après la SAISIE d'un code.

d - CONTROLE

Objet : - Détecter l'existence ou la non-existence d'une réalisation d'entité. Si l'existence est détectée et que le contrôle est négatif, alors on garde l'adresse dans un Xi.

- Edition d'un message dans le cas où le contrôle est positif.

Syntaxe :

contrôle { pas / existe } NOM-ENTITE X_i Z_i NN

Remarques :

(1) NOM-ENTITE = est le nom de l'entité dont on va contrôler l'existence ou non d'une réalisation ayant pour code Z_i.

(2) X_i = variable qui contiendra l'adresse de la réalisation de l'entité si on fait un contrôle existe et que celle-ci est trouvée.

(3) Z_i = variable qui contient le code de la réalisation de l'entité à contrôler.

(4) NN = N° du message à éditer si on fait :
- contrôle existe et que la réalisation est trouvée
- contrôle pas et que la réalisation n'existe pas.

Schéma de programme :

```
defmac CONTROLE &1 &2 &3 &4 &5
    m Y23 = u
    si &4 ¬= u alors
(1)     m Y23 = numde un &2 &3 avec code = &4 ;
    fin
    si &1 y23 alors
        ERREUR &5
        m &3 = u
    fin
fdef
```

COMMENTAIRES

(1) En sortie de la macro, la variable Y23 contient le numéro d'ordre de la réalisation de l'entité :

- 0 si la réalisation n'existe pas
- n° si la réalisation existe.

&3 contient l'adresse de &2.

(2) Macro d'édition de l'erreur N° &5.

Exemple d'utilisation

```
SAISIE CODE Z3 99
CONTROLE PAS ACTION X2 Z3 25
```

Dans X2, on a l'adresse de l'action si elle existe, sinon le message 25 est édité.

e - CONTROLE-PLAGE

Objet : Contrôler les bornes d'une valeur numérique.

Syntaxe : contrôle-plage VAL INF SUP NN

REMARQUES :

① VAL = valeur à contrôler dans un Yi

② INF = limite inférieure

③ SUP = limite supérieure

④ NN = N° du message à éditer si la valeur est en dehors des bornes.

Schéma de programme :

```
          defmac CONTROLE-PLAGE &1 &2 &3 &4
              si &1 = u ou &1 < &2 ou &1 > &3 alors
(1)                   ERREUR &4
              fin
          fin
```

COMMENTAIRE :

(1) Macro d'édition d'erreur N° &4.

Exemple d'utilisation :

```
SAISIE NATURE Z1 54
m Y1 = Z1
CONTROLE-PLAGE Y1 1 10 55
```

f - ERREUR

Objet : Edition du libellé d'un message. Si on travaille en option bavarde alors édition de l'explication associée au message après confirmation de l'utilisateur.

Syntaxe : ERREUR NN

Remarque : NN = code du message à éditer.

Schéma de programme :

```
        defmac ERREUR &1
                m Y1 = numde un MESSAGE avec code = &1
          faire
(1)             si Y1 = u alors i 'message inconnu' sortie fin
(2)             i LIBELLE de un MESSAGE Y1
(3)             si Y21 = 1 alors sortie fin
             faire
(4)             SAISIE EXPLICATION Z7 0
                si Z7 = u ou Z7 ¬= 'OUI' ou Z7 ¬= 'NON' alors
                    i 'Répondre OUI ou NON SVP' refaire
                fin
             fin
(5)             si Z7 = u ou Z7 = 'NON' alors sortie fin
(6)             i DEFINITION de un MESSAGE Y1
          fin
        fdef
```

COMMENTAIRES

(1) Si le message N° &1 n'est pas enregistré dans la base, on sort de la macro ERREUR après édition de 'message inconnu'.

(2) Edition du libellé du message N° &1.

(3) Si on travaille en option silencieuse, alors on sort de la macro.

(4) Demande à l'utilisateur s'il veut une explication sur l'erreur.

(5) Si l'utilisateur ne répond rien ou bien s'il répond 'non', on sort de la macro.

(6) Edition de l'explication associée au message N° &1.

Exemple d'utilisation : (voir exemple d'utilisation de la macro SAISIE).

g - INIT

Objet : Lorsqu'un opérateur de MACSI-1 (les commandes de MACSI-1 pour les utilisateurs) doit déclarer ou doit accéder à une réalisation d'entité, les mises à jour et contrôles suivants sont réalisés par INIT :

- Si le code n'existe pas dans le DICO, il est créé ainsi que la réalisation de l'entité correspondante (CODE et LIBELLE, TOPVAL = 2)
- Si le code existe dans le DICO, l'accès à la réalisation de l'entité est autorisée après vérification des droits de l'utilisateur.

A la sortie de la macro INIT, l'adresse de la réalisation de l'entité affectée est récupérée dans une variable Xi.

Cette macro INIT est certainement l'outil principal de l'analyse descendante ; elle permet l'initialisation du "TOPVAL" d'un composant.

Syntaxe : init NOM-ENT Xi NAT Zi

Remarques :

(1) NOM-ENT = nom de l'entité

X_i = Variable de type X qui pointe sur la réalisation de l'entité en sortie de INIT

NAT = Code nature de l'entité. Ce code est le suivant :

1 = Service administratif
2 = Groupe d'exécution
3 = Personne
4 = Module
5 = Action
6 = Commande
7 = Dicinit
8 = Opérateur
9 = Message
10 = Réclamation
11 = Fichier
12 = Document

20 = Enregistrement

21 = Ensemble-relation

22 = Propriétés

23 = Structure Socrate associée

Z_i = Variable qui contient le code interrogé.

Schéma de programme :

```
        defmac INIT &1 &2 &3 &4
            si existe un DICO X8 avec code = &4
            alors
                faire
                    si pas un &1 &2 avec code = &4 alors
(1)                         ERREUR 14 sortie
                    fin
                    m Y23 = TOPVAL de &2
                    si y23 = 1 ou y23 = 3 ou y23 = 5 alors
(2)                         ERREUR 15 sortie
                    fin
                    si y23 = 2 ou y23 = 4 ou y23 = 6 alors
                      m &4 = INITIALISATEUR DE X8
                      si VOTRE-CODE de un DICINIT Y22 ¬= &4 alors
(3)                         ERREUR 16 sortie
                      fin
                    fin
                    si PROJET de &2 ¬= Y24 alors
(4)                         ERREUR 17 sortie
                    fin
(5)                 i LIBELLE de &2
(6)                 i 'D'ACCORD ?' m Z2 = EXT
                    si Z2 = 'NON' alors m &2 = u
                          COMMENTAIRE 58
                    fin
                fin
            sinon
```

```
(7)                 bloquer
(8)                     g un DICO X8
(9)                     g un &1 &2
(10)                    m CODE de X8 = &4
(11)                    m CODE de &2 = &4
(12)                    m TOPVAL de &2 = 1

                    libérer
(13)                    m NATURE de X8 = &3
(14)                    m PROJET de X8 = Y24
(15)                    m PROJET de &2 = Y24
(16)                    m LIBELLE de &2 = ext

                    bloquer
(17)                    m TOPVAL DE &2 = 2
(18)                    m INITIALISATEUR de X8 = VOTRE-CODE de un DICINIT Y22

                    libérer
(19)                    g un CODCREE X3 de un DICINIT Y22
(20)                    m CODE de X3 = X8
            fin
```

Commentaires :

(1) Message d'erreur indiquant l'existence du code dans DICO mais la non-existence de la réalisation de l'entité.

(2) Message d'erreur indiquant que l'utilisateur n'est pas l'initialisateur de la réalisation de l'entité car une autre personne au même moment est en train de la déclarer, la mettre à jour ou la modifier.

(3) Message d'erreur car l'utilisateur essaie une opération sur une réalisation d'entité qui n'a pas été déclarée par lui.

(4) Message d'erreur indiquant la tentative d'accès à une réalisation d'entité qui ne correspond pas au projet de l'utilisateur.

(5),(6) On édite le libellé de la réalisation de l'entité pour demander confirmation à l'utilisateur. S'il est d'accord, l'opération est valide : on sort de la macro INIT avec une variable X_i (&2) qui pointe vers la réalisation de l'entité.

(7) L'opération BLOQUER interdit tout accès à d'autres utilisateurs aux caractéristiques qui sont dans le bloc.

(8),(9),(10),(11) Le DICO et l'entité correspondante sont créés ainsi que leurs codes mis à jour.

(12) Le TOPVAL de la réalisation de l'entité est mis à 1.

(13) La nature du DICO correspondant est mise à jour.

(14),(15) Le "PROJET" du DICO et de la réalisation de l'entité est mis à jour avec le code du projet en cours gardé dans la variable Y24.

(16),(17),(18) Le libellé de la réalisation de l'entité est demandé, le TOPVAL passe à 2 et l'INITIALISATEUR est mis à jour.

(19),(20) Un CODCREE est généré et son CODE mis à jour.

h - TRAITE

Objet : Utilitaire de demande du type de traitement à effectuer. La réponse doit être :

- Oui (validation des opérations précédentes)
- Non (annulation des opérations précédentes)

ou - Résumé (récapitulation des opérations précédentes)

L'utilitaire fait une redemande tant que la réponse n'est pas l'une des trois ci-dessus mentionnées. Il récupère la réponse correcte dans la variable Z_i passée comme paramètre.

Syntaxe : traite Z_i

i - GEX

Objet : Programme de génération et remplissage d'une explication. A la sortie la variable X9 pointe sur l'explication créée (X9 sera indéfini si aucun texte n'est donné).

Syntaxe : gex

Remarques :

(1) Une fin de texte est délimitée par une étoile (*) comme dernier caractère d'une ligne.

(2) La tabulation à 60 caractères est obtenue en mode bavard.

(3) Le numéro ou code du texte de l'explication sera édité après la création (ce numéro est généré par MACSI-1 et communiqué à l'utilisateur).

j - ACCESMAJ

Objet : Programme à utiliser dans les opérateurs de mise à jour. Ses fonctions sont les suivantes :

- Saisie du code de la réalisation de l'entité à mettre à jour
- Vérification de la possibilité de mise à jour (TOPVAL)
- Vérification du droit d'accès (INITIALISATEUR, PROJET)
- Edition du libellé de la réalisation de l'entité.

En sortie du programme :

- La variable X1 pointe sur la réalisation de l'entité à mettre à jour
- La variable X8 pointe le DICO correspondant
- Le TOPVAL de la réalisation de l'entité est mis à 3
- La variable Z1 contient le code de la réalisation de l'entité à mettre à jour.

Syntaxe : accesmaj NOM-ENT

NOM-ENT : Nom de l'entité d'appartenance de l'objet à mettre à jour.

Schéma de programme :

```
          defmac ACCESMAJ &1
           faire
(1)             SAISIE CODE Z1 21
                NORMCOD Z1
(2)             CONTROLE PAS &1 X1 Z1 22
                si PROJET de X1 ¬ = Y24 alors
(3)                ERREUR 35 sortie
                fin
                si TOPVAL de X1 ¬ = 2 alors
(4)                ERREUR 23 sortie
                fin
(5)             m X8 = un DICO avec CODE = Z1 ;
                si INITIALISATEUR de X8 ¬ = VOTRE-CODE DE UN DICINIT Y22 alors
(6)                ERREUR 16
                fin
                m TOPVAL de X1 = 3
(7)             i LIBELLE de X1
(8)             m CODE-EN-COURS de un DICINIT Y22 = Z1
(9)
          fdef
```

Commentaires :

(1) Saisie et normalisation du code de l'entité à mettre à jour.

(2) Contrôle de l'existence de la réalisation de l'entité.

(3) Contrôle de cohérence entre les PROJET.

(4) Contrôle du TOPVAL.

(5) Changement dans X8 de l'adresse du DICO correspondant.

(6) Contrôle du droit d'accès par rapport à l'INITIALISATEUR.

(7) Mise à jour du TOPVAL et édition du libellé.

(8) Mise à jour du CODE-EN-COURS.

(9) La macro ACCESMAJ ouvre une boucle faire qui est fermée par la macro FINMAJ (cf. p. 4.33.) à la fin d'un opérateur de mise à jour.

k - ACCESCRE et ACCESMOD

Objet : Ce sont deux programmes équivalentes à ACCESMAJ mais utilisés dans les opérateurs de création et de modification.

l - FINMAJ

Objet : Programme qui "ferme" un opérateur de mise à jour. Ses fonctions sont les suivantes :

- Gestion du topval final (6 ou 8)
- Edition des codes déclarés dans l'opération de mise à jour
- Stockage de ces codes dans CODES-A-LISTER
- Effacement de CODE-EN-COURS
- Ferme le bloc faire ouvert par ACCESMAJ.

Syntaxe : finmaj NOM-ENT

NOM-ENT : nom de l'entité dont on sort.

Schéma de programme :

```
        defmac FINMAJ &1
            faire
                SAISIE ACCES Z6 29
(1)                 si Z6 = u ou (Z6 ¬= "public" et Z6 ¬= "privé") alors
                        ERREUR 29 refaire
            fin
            si existe un CODCREE de un DICINIT Y22 alors
               i 'Vous venez d'initialiser :'
(2)            Pour tout CODCREE X1 de un DICINIT Y22
                 i code de X1
               fin
            fin
            si Z6 = 'privé' alors m TOPVAL de X1 = 6
(3)                         sinon m TOPVAL de X1 = 8
            fin
(4)         m CODE-EN-COURS de un DICINIT Y22 = u
            g un CODE-A-LISTER X3
(5)         m CODE de X3 = CODE de X1
(6)       fin
        fdef
```

Commentaires :

(1) Demande du type d'accès (TOPVAL) pour la réalisation qui vient d'être mise à jour.

(2) Edition de la liste de tous les éléments initialisés par la mise à jour.

(3) Mise à jour du TOPVAL selon l'accès.

(4) Effacement du CODE-EN-COURS.

(5) Mise à jour d'un CODE-A-LISTER.

(6) Fermeture du bloc faire ouvert par la macro ACCESMAJ.

m - FINCRE, FINMOD

Objet : Ces deux programmes ont une fonction équivalente à FINMAJ pour les opérations de création et modification.

n - Citons brièvement d'autres utilitaires :

Identificateur	Libellé
APPEL	Envoi de messages entre consoles.
CONTREX	Contrôle d'existence d'une caractéristique de type mot.
EFFACE	Suppression de toutes les entités initialisées et des DICO correspondants par une création, mise à jour ou modification.
GLED	Génération d'un lien EXPLICATION-DICO.
SLED	Suppression d'un lien EXPLICATION-DICO.

4.d.2. Utilisation des programmes de service pour l'implémentation des programmes conversationnels

En 4.d.1., nous avons décrit les principaux opérateurs de service utilisés pour l'écriture des opérateurs de MACSI-1.

Nous nous proposons maintenant de décrire la logique des traitements de quelques-uns de ces opérateurs. Nous utiliserons l'opérateur de MACSI-1 analyse (cf. Chapitre 3) pour spécifier les schémas de programme de ces opérateurs. Ces schémas seront décrits à l'aide des opérateurs de service et du langage de requêtes Socrate comme nous l'avons décidé au chapitre 3.

Nous ferons les commentaires nécessaires à la suite des schémas des programmes pour garder une présentation plus claire.

Les programmations définitives ainsi que des utilisations de ces opérateurs figurent en annexes B et C.

Nous prendrons 3 opérateurs du mode conversationnel de MACSI-1 :

- un opérateur de création (creens)
- un opérateur de mise à jour (majco)
- un opérateur de modification (modpp).

Les programmes pour le mode batch seront introduits dans § 4.e.

4.d.2.1. L'opérateur majco

a) Implémentation

Au Chapitre 1 (p. 1.87) nous avons présenté le graphe de description d'une mise à jour d'une commande (MAJCO).

Les spécifications de MAJCO sont décrites avec la commande analyse :

```
analyse  MAJCO
     type 1
     accès 3
     mode 1
     appel ACCESMAJ
           SAISIE
           CONTROLE-PLAGE
           NORMCOD
           INIT
           TRAITE
           EDICO
           FINMAJ

specif *
(1)        defpro [MAJCO]

(2)              ACCESMAJ COMMANDE
(3)                       SAISIE TYPECO Z2 300
                                m Y1 = Z2
(4)                       CONTROLE-PLAGE Y1 1 99 301

                          i 'DOCUMENT :'
(5)                       faire SAISIE CODEDO Z2 302
(6)                             NORMCOD Z2 Z3
                                si Z2 = '$$' alors sortie fin
(7)                             INIT DOCUMENT X3 12 Z2
                                refaire
                          fin
(8)                       faire TRAITE Z7
(9)                             si Z7 = 'RESUME' alors EDICO ... refaire fin
                          fin
(10)                      si Z7 = 'NON' alors /* annulation ... */
                                                sortie
                          fin
(11)                         /* mise à jour effective de la base */
(12)             FINMAJ COMMANDE
           fdef
*
fanalyse
```

Ceci décrit "l'analyse" de l'opérateur majco chargé de saisir des informations (type 1), il accède à la base en lecture et écriture (accès 3) et il est utilisé en mode conversationnel (mode 1) et fait appel à des opérateurs de service (appel : ACCESMAJ, SAISIE, ...).

b) Commentaires

(1) Tout opérateur de MACSI-1 sera programmé sous forme d'un programme précompilé Socrate.

(2) L'opérateur ACCESMAJ permet ici de saisir le code (CODECO) d'une commande, d'en vérifier les droits d'accès (TOPVAL), de lire la DEFINITION si l'accès était permis sinon de générer une erreur suivie d'une sortie du bloc ACCESMAJ-FINMAJ.

(3) L'opérateur SAISIE fait la lecture du type (TYPECO) de la commande dans la variable Z2 avec édition éventuelle de l'explication 300.

(4) CONTROLE-PLAGE vérifie que TYPECO est un entier compris entre 1 et 99 et en cas d'erreur édite le message 301.

(5) et (6) Ici nous retrouvons la boucle faire ... refaire fin permettant d'associer plusieurs documents à la commande en mise à jour. Il y a la saisie (SAISIE) et la normalisation (NORMCOD) du code (CODEDO) d'un document. La sortie de cette boucle est exécutée lorsqu'il y a saisie d'un CODEDO = 'ØØ'.

(7) L'opérateur INIT vérifie l'état du document de code Z2 (= CODEDO) :

- si c'est un nouveau document (qui n'a jamais été cité), INIT se charge de la gestion du DICO de nature 12, du TOPVAL, de la saisie du libellé (LIBELDO).
- sinon INIT calcule seulement l'adresse X3 de ce document existant.

(8),(9),(10),(11) L'utilisateur, par l'opérateur TRAITE, pourra dicter maintenant ce qu'il faut faire des informations qu'il vient de rentrer :

- il peut demander une récapitulation (Résumé) du contenu de cette commande qu'il vient de mettre à jour
- il peut annuler (Non) tout ce qu'il a fait depuis qu'il est sous le contrôle de l'opérateur majco
- ou bien il est d'accord (Oui) avec toutes les descriptions qu'il vient de donner alors le système se charge de la mise à jour effective de la base

(12) L'opérateur FINMAJ marque la fin de la mise à jour de la commande de code = CODECO et détermine si elle sera en accès "public" ou "privé".

c) Remarques

- Nous pouvons rappeler le rôle de la zone de travail (DICINIT) qui permet de stocker toutes les informations données par l'utilisateur d'un opérateur MACSI-1 (par exemple majco), de faire des contrôles de validité ou de cohérence dans les descriptions, de mettre à jour réellement la base seulement à la fin de l'exécution de l'opérateur concerné et après avis de cet utilisateur (cf. macro TRAITE).

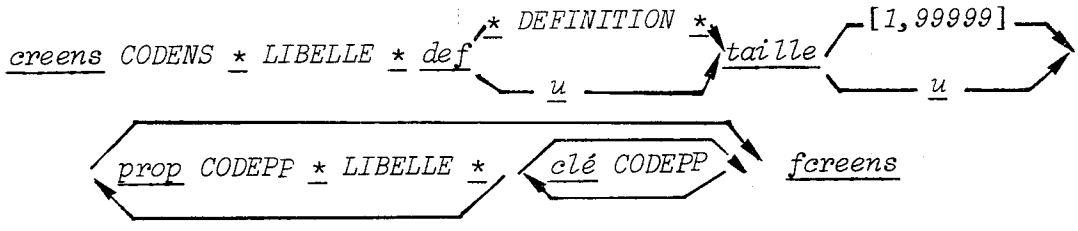
- Il est bien évident qu'ici nous n'avons présenté que le schéma de programme relatif à l'opérateur majco, et que la programmation réelle d'un tel opérateur devra à partir de ce schéma, le compléter pour résoudre tous les problèmes soulevés en cas de saisie d'informations erronées, d'une redemande de saisie, ... Il faut aussi définir les variables Xi employées, mettre à jour les chaînes d'inverse ...

- Mais il est bon de souligner que l'utilisation des opérateurs de service (SAISIE, INIT, ...) permet une programmation structurée des opérateurs de MACSI-1 qui dépasse très rarement le nombre de 30 ou 40 requêtes Socrate élémentaires ou insertion de macro-instructions pour chacun.

4.d.2.2. L'opérateur creens

a) Implémentation

Nous allons donner ici un exemple d'implémentation de l'opérateur de création d'un ensemble (CREENS) en mode conversationnel.



analyse CREENS

type 1
accès 3
mode 1
appel COMMENTAIRE
ACCESRE
GEX
SAISIE
CONTROLE-PLAGE
NORMCOD
INIT
TRAITE
FINCRE

```
specifs *
(1)     defpro CREENS

(2)         COMMENTAIRE 525
(3)         ACCESCRE ENS-REL 21
(4)                 m TYPE de X1 = 1
                    i 'DEFINITION :'
(5)                 EXEC GEX

                    i 'TAILLE :'
(6)                 SAISIE TAILLE Z2 0
                        m Y2 = Z2
(7)                 CONTROLE-PLAGE Y2 1 99999 579

                    i 'PROPRIETES :'
(8)                 faire SAISIE CODEPP Z2 0
(9)                     NORMCOD Z2 Z3
                        si Z2 = '$$' alors sortie fin

(10)                    INIT PROPRIETE X2 22 Z2

                        refaire
                    fin
                    i 'CLES :'
(11)                faire SAISIE CODEPP Z2 0
(12)                    NORMCOD Z2 Z3
                        si Z2 = '$$' alors sortie fin
(13)                      /* vérification que CODEPP est une propriété
                             de l'ensemble */
                        refaire
                    fin

(14)                TRAITE /* Récapitulation, annulation ou validation *

(15)        FINCRE ENS-REL
        fdef
* fanalyse
```

b) Commentaires

(1),(2) L'opérateur COMMENTAIRE permet l'édition (en "mode bavard" uniquement) de l'explication 525 qui indiquera à l'utilisateur la définition de l'opérateur CREENS et ses normes d'utilisation.

(3) L'opérateur ACCESCRE saisit le code (CODENS) d'un composant ENS-REL (de nature 21) ; il vérifie qu'aucun autre composant déjà dans la base possède ce même code, sinon une erreur est imprimée et une sortie du bloc ACCESCRE-FINCRE est générée.
Si l'accès à ce composant est déclaré bon, il y a saisie de son libellé et gestion de son "DICO".

(4) Un ensemble de base est de type 1 (2 = une relation).

(5) L'opérateur GEX permet de lire un texte de 100 lignes maximum de 60 caractères.

(6),(7) La taille d'un ensemble est un entier positif.

(8),(9),(10) La boucle faire ... refaire fin permet d'enregistrer toutes les propriétés d'un ensemble et de les initialiser (INIT).

(11),(12),(13) Lecture des clés de l'enregistrement avec vérification que ces clés sont elles-mêmes des propriétés de l'ensemble.

(14) Ici, nous retrouverons l'emploi de l'opérateur de service TRAITE permettant de récapituler, puis d'annuler ou de valider la séquence de création d'un ensemble.

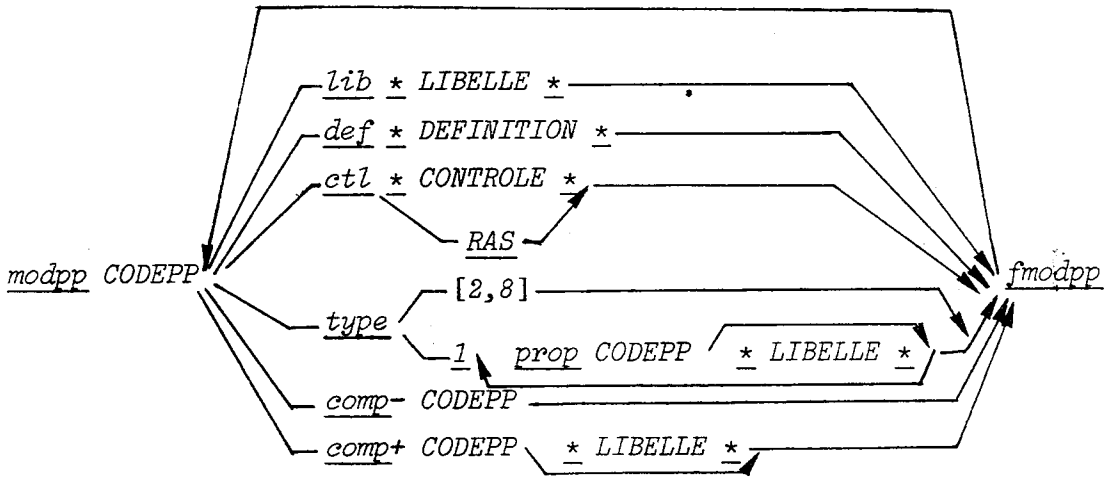
(15) FINCRE définit l'accès public ou privé à cet ensemble.

4.d.2.3. L'opérateur MODPP

a) Implémentation

Dans tout ce qui précède, nous n'avons pas donné de règle de description permettant de modifier ou supprimer des composants : nous avons seulement présenter des règles de description autorisant des créations ou des mises à jour. Il est bien évident qu'à tout composant on doit pouvoir associer une commande de modification et éventuellement de suppression.

Introduisons la commande de modification des spécifications d'une propriété (MODPP), par son graphe de description.



Naturellement, nous aurions pu décrire cette nouvelle commande par majco, mais donnons simplement ses spécifications avec la commande analyse.

```
analyse   MODPP

    type 1
    accès 3
    mode 1
    appel COMMENTAIRE
          ACCESMOD
          SAISIE
          NORMCOD
          EDIPP
          MOLIB
          MODEF
          MOCTL
          MOTYPP
          MOCPP
          MOCPM
          FINMOD
          ERREUR
```

```
specifs *
        defpro [MODPP]
(1)             COMMENTAIRE 510
(2)             ACCESMOD PROPRIETE
                    faire SAISIE MODIF-DE Z2 0
(3)                         si Z2 = U alors refaire fin
                            NORMCOD Z2 Z3
(4)                         si Z2 = '$$' alors sortie fin
                            si Z2 = 'RECAP' alors EDIPP ... refaire fin
                            si Z2 = 'LIB' alors MOLIB refaire fin
(5)                           :    'DEF'   :   MODEF    :     :
                              :    'CTL'   :   MOCTL    :     :
                                   'TYPE'      MOTYPP
                                   'COMP+'     MOCPP
                                   'COMP-'     MOCPM
(6)                         ERREUR 595
                            refaire
                    fin
(7)             FINMOD PROPRIETE
        fdef
* fanalyse
```

b) Commentaires

(1) Le commentaire 510 permettra d'indiquer à l'utilisateur quelles sont les modifications permises sur une propriété et comment les décrire.

(2) L'opérateur ACCESMOD permet après avoir saisi un code de vérifier que c'est bien un code de propriété, et que la propriété citée a un topval (= 4 ou 6) qui autorise des modifications.

(3) Saisie et normalisation du type de la modification à effectuer.

(4) L'utilisateur a la possibilité de demander une récapitulation des spécifications de la propriété concernée, ou de dire qu'il a terminé de faire des modifications ($$).

(5) Nous permettons 6 sortes de modifications d'une propriété concernant :
- le libellé (lib)
- la définition (déf)
- la description des contrôles (ctl)

- le type (entier, réel, mot, composé, ...) (type)
- l'ajout d'un composant à une propriété composée (comp+)
- la suppression d'un composant à une propriété composée (comp-)

(6) L'erreur 595 sera éditée en cas de lecture d'un code de modification interdit.

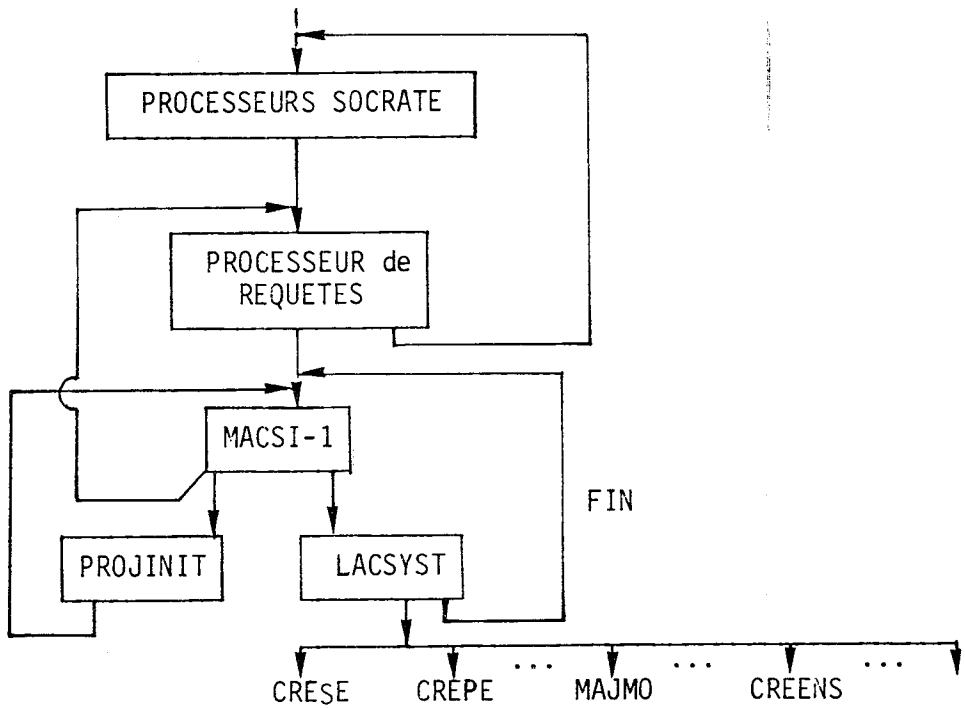
(7) FINMOD ferme l'accès en modification à cette propriété et redéfinit l'accès public ou privé.

4.d.3. Le programme principal MACSI-1

4.d.3.1. Accès à MACSI-1

Nous avons déjà décrit les spécifications fonctionnelles de ce module MACSI-1 dans le chapitre 2. Notre but ici est uniquement de vous montrer un schéma possible d'implémentation que nous avons volontairement très simplifié.

Nous avons indiqué en 4.b.5. le schéma d'appel de MACSI-1 par rapport au système Socrate ; nous allons en détailler la partie inférieure.



Remarques :

- Nous ne décrirons pas l'opérateur PROJINIT ; donnons simplement sa définition : il permettra aux administrateurs des bases MACSI-1 de gérer les différents projets :
  . ajout ou suppression de spécificateurs à un projet
  . description ou modification de la liste des opérations autorisées pour une personne dans un projet
  . initialisation d'un nouveau projet
  . ...

4.d.3.2. L'opérateur LACSYST

Décrivons cet opérateur avec le même formalisme qu'en § 4.d.2.

a) Implémentation

analyse LACSYST

type 1
accès 3
mode 1
appel SAISIE
NORMCOD
CONTROLE
ERREUR
CREPE
CRESE
MAJPE
MAJCO
:

```
spécifs *

    defpro [LACSYST]

        faire SAISIE VOTRE-CODE Z1 1
              NORMCOD Z1
(1)           CONTROLE PAS PERSONNE X1 Z1 2
                    /* si un imposteur s'introduit dans la base alors
                       sortie */

(2)           SAISIE PROJET Z2 3
              NORMCOD Z2
                /* vérification que la personne ci-dessus peut travaille
                   sur ce projet */
                /* affectation d'une zone de travail (dicinit)     */
                /* initialisation du dicinit : date, ...           */

              faire SAISIE OPERATION Z3 4
                    NORMCOD Z3
                    CONTROLE PAS COMMANDE X3 Z3 4
(3)
                    /* contrôle que cette opération est autorisée pour
                       la personne citée, dans le cadre du projet
                       mentionné */

                    si Z3 = 'FIN' alors sortie fin
                    si Z3 = 'CREPE' alors exec CREPE refaire fin
                    si Z3 = 'CRESE' alors exec CRESE refaire fin
                    si Z3 = 'MAJPE' alors exec MAJPE refaire fin
                    si Z3 = 'MAJGR' alors exec MAJGR refaire fin
                    si Z3 = 'MAJMO' alors exec MAJMO refaire fin
                    si Z3 = 'MAJCO' alors exec MAJCO refaire fin
(4)                 si Z3 = 'CREENS'alors exec CREENS refaire fin
                        :
                    si Z3 = 'MODPP' alors exec MODPP refaire fin
                    si Z3 = 'MAJEG' alors exec MAJEG refaire fin
(5)                 ERREUR 111
                    refaire

              fin

            /* fin de session, libération du dicinit, ...      */
        fin
fdef
```

b) Commentaires

(1) Après une saisie et une normalisation du mot de passe d'une personne (VOTRE-CODE), le système vérifie que cette personne est autorisée à travailler sur la base ; sinon une requête sortie renvoie l'utilisateur au niveau Socrate.

(2) La saisie et la normalisation du code projet (PROJET) permet de vérifier que la personne peut effectivement travailler sur le projet.

(3) Maintenant, cette personne peut diriger des exécutions d'opérations sur ce projet. A chaque opération demandée, il y aura naturellement une vérification du droit de faire cette opération par cette personne pour ce projet.

(4) Si l'opération est reconnue comme utilisable (3) alors il y a appel de l'opérateur correspondant, exécution de cet opérateur, et redemande d'une nouvelle opération.

(5) Si l'opération est inconnue dans MACSI-1, alors édition du message d'erreur 111.

4.e. Gestion des automates d'états finis associés aux règles de description

4.e.1. Introduction

Les spécifications d'un projet sont données en utilisant un formalisme défini par les Règles de Description. Ces règles de description, que nous avons présentées aux chapitres précédents, sont, en fait, des automates déterministes d'états finis exprimant l'ordre dans lequel sont introduites les valeurs des propriétés et relations associées à un type de composant.

Cette remarque offre des possibilités qui sont, à notre avis, les plus importantes pour la conception d'un système flexible et évolutif. En effet, aux automates déterministes d'états finis, on peut associer un interpréteur unique des spécifications qui fera appel à des programmes très simples de contrôle syntaxique, sémantique et d'exécution.

Alors, l'ajout d'une nouvelle règle de description, qui pourrait être une opération compliquée, est limitée aux opérations suivantes :

* L'enregistrement en mémoire de cette règle (cf. § 4.e.3.1.).
* La gestion des programmes standards de contrôle syntaxique.
* L'écriture des programmes de contrôle sémantique et de mise à jour de la base à l'aide des programmes "utilitaires" catalogués.

Précisons ces différents points.

Le lecteur informaticien reconnaîtra dans les règles de LACSYST une structure d'automates d'états finis. En effet, si par exemple, on prend la règle majco :

on peut lui associer un automate d'états finis déterministe A avec :

$A = (V_T, Q, q_1, F, \mu)$

V_T = {majco, CODECO, *, DEFCO, type, TYPCO, doc, CODEDO, LIBELDO, fmajco}

$Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}\}$

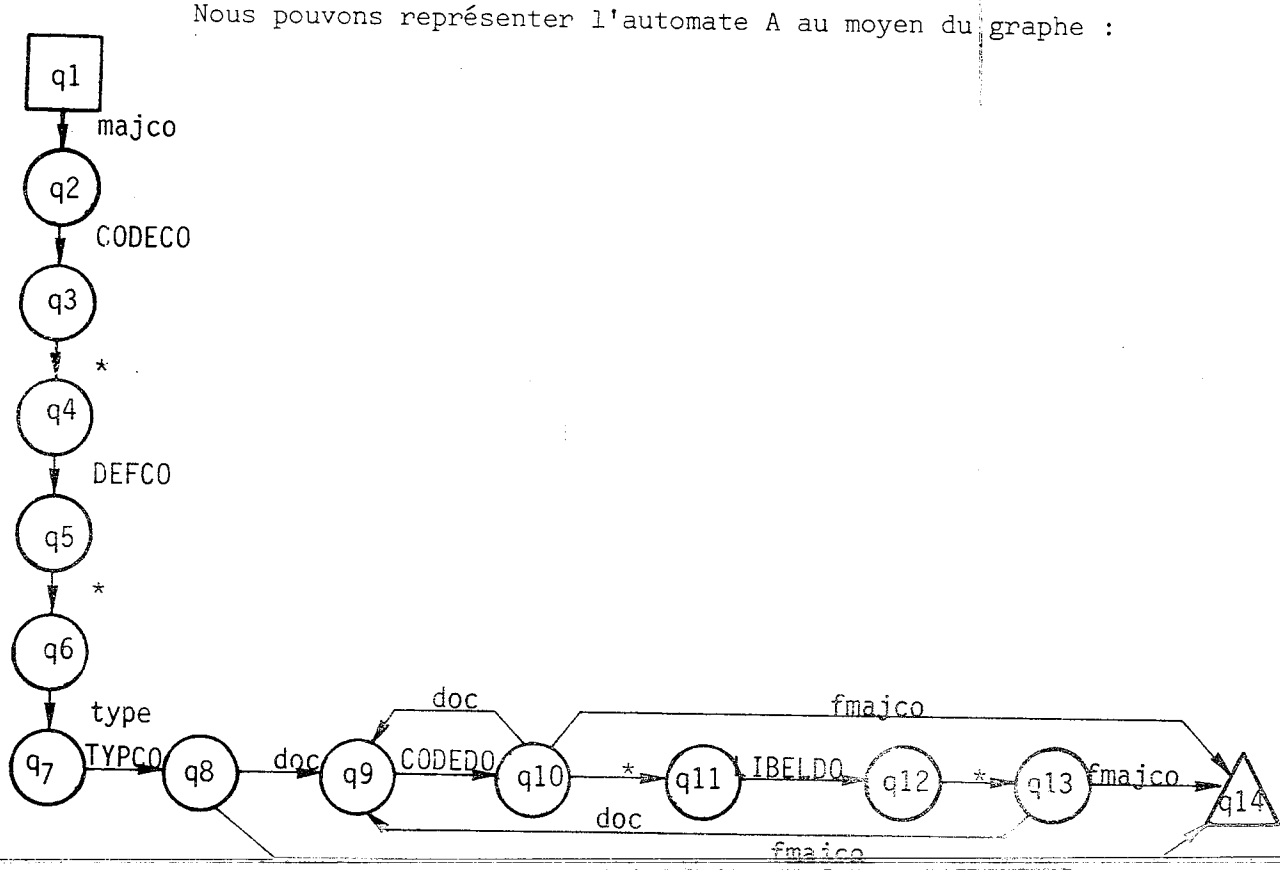
q_1 = état initial

$F = \{q_{14}\}$ ensemble des états finals

μ = l'application $Q \times V_T \rightarrow Q$

q \ σ	majco	CODECO	*	DEFCO	type	TYPCO	doc	CODEDO	LIBELDO	fmajco
q1	q2									
q2		q3								
q3			q4							
q4				q5						
q5			q6							
q6					q7					
q7						q8				
q8							q9			q14
q9								q10		
q10			q11				q7			q14
q11									q12	
q12			q13							
q13							q9			q14
q14										

Nous pouvons représenter l'automate A au moyen du graphe :



Nous savons qu'une grammaire d'états finis, ou de Kleene, peut être associée aux automates d'états finis (cf. VAUQUOIS [33]). En effet, étant donné A (automate fini déterministe), on se propose de trouver une grammaire G de type 3 ou de Kleene telle que

$L(G) = L(A)$

L(G) = langage décrit par la grammaire

L(A) = langage décrit par l'automate.

Alors on définit G de la manière suivante :

Si $A = (V_T, Q, q_1, F, \mu)$

on a : $G = (V_T, Q, q_1, P)$

où les règles de production P sont :

1°) $B \rightarrow aC \in P$ si $\mu(B,a) = C$

2°) $B \rightarrow a \in P$ si $\mu(B,a) = C$ et $C \in F$

Aussi, nous pouvons donc associer à LACSYST un langage constitué par une famille de grammaires de Kleene.

4.e.2. Concepts de base pour l'analyse d'une phrase LACSYST

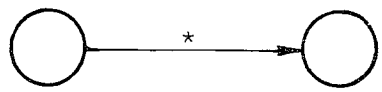
Nous allons alors représenter l'ensemble des règles de production de la grammaire LACSYST par des "diagrammes de transition". Le lecteur retrouvera une présentation didactique de ces concepts dans [6], [19]. L'ensemble des diagrammes et de l'algorithme qui les exploite constitue ce que nous appellerons l' "analyseur". Nous entendrons aussi par "fenêtre" le symbole courant de la chaîne source que l'on est entrain d'analyser.

Un métasymbole de la grammaire est défini par l'ensemble des règles de production dans lesquelles il apparaît en partie gauche. A chaque métasymbole, correspondra un diagramme de transition de même nom représentatif de cet ensemble de règles de production. Un diagramme de transition est donc un graphe orienté, connexe, dont les arcs représentent soit des symboles terminaux, soit des métasymboles ; chaque chemin de ce graphe représente une des règles de production définissant le métasymbole considéré.

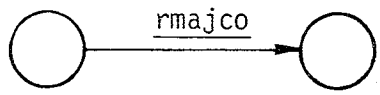
Chaque diagramme aura un noeud d'entrée vers lequel ne converge aucun arc, et un noeud de sortie duquel ne diverge aucun arc.

Les différents arcs constituant un diagramme sont de trois sortes :

1) Les arcs terminaux : ils représentent les symboles terminaux. On note le symbole à côté de l'arc :

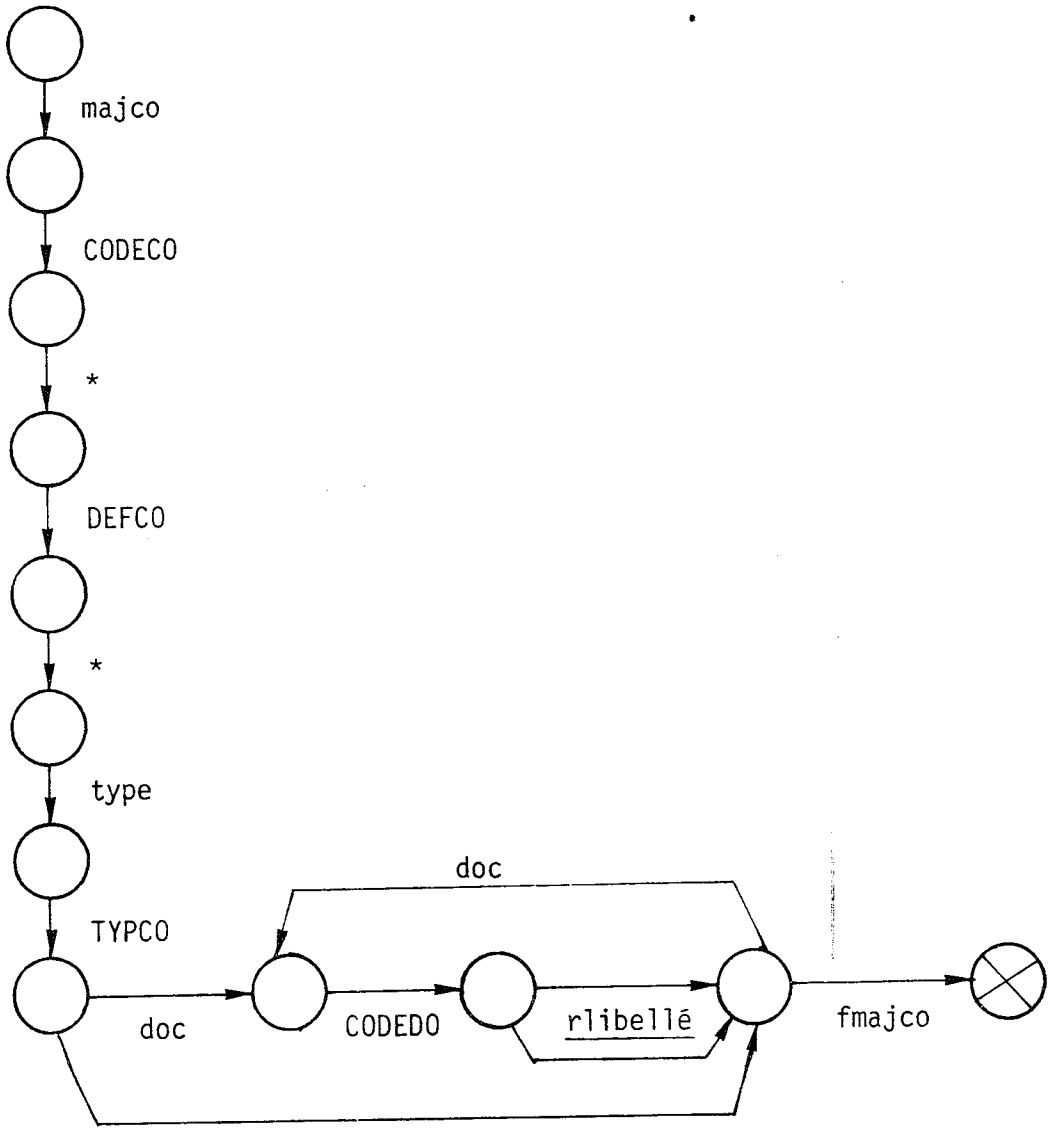


2) Les arcs non-terminaux : ils représentent les symboles non terminaux. Le nom est noté, souligné, à côté de l'arc :



3) Les arcs blancs : ils jouent un rôle important dans la construction des diagrammes. Ils représentent la chaîne vide. On le notera :

Donnons l'exemple du diagramme de transition représentant le métasymbole <RMAJCO> :



La reconnaissance d'une sous-chaîne du langage se concrétisera par le passage d'un noeud à un autre en suivant l'arc qui les relie. Une telle traversée aura lieu :

- pour un arc terminal, lorsque le symbole qu'il représente et le symbole de la chaîne source coïncideront,

- pour un arc non terminal, par la traversée du diagramme de même nom, c'est-à-dire par la recherche d'un chemin allant du noeud d'entrée au noeud de sortie de ce diagramme,

- pour un arc blanc, lorsque tous les arcs non blancs divergeant du même noeud que lui ne permettront pas la traversée.

Ainsi, la traversée d'un diagramme de transition sélectionne une séquence de symboles (terminaux ou non), dont la concaténation constitue la partie droite d'une règle de production. Cette traversée revient donc à remplacer une sous-chaîne de la chaîne analysée par le métasymbole défini par le diagramme balayé.

S étant l'axiome de départ de la grammaire, il y aura un diagramme de départ S, l'analyse consistant à balayer ce diagramme. Lorsque toute la chaîne source a été analysée, deux cas peuvent se présenter :

- On se trouve sur le noeud de sortie du diagramme de départ. Dans ce cas, la chaîne source a permis de sélectionner un chemin du diagramme de départ : nous avons donc une phrase syntaxiquement correcte.

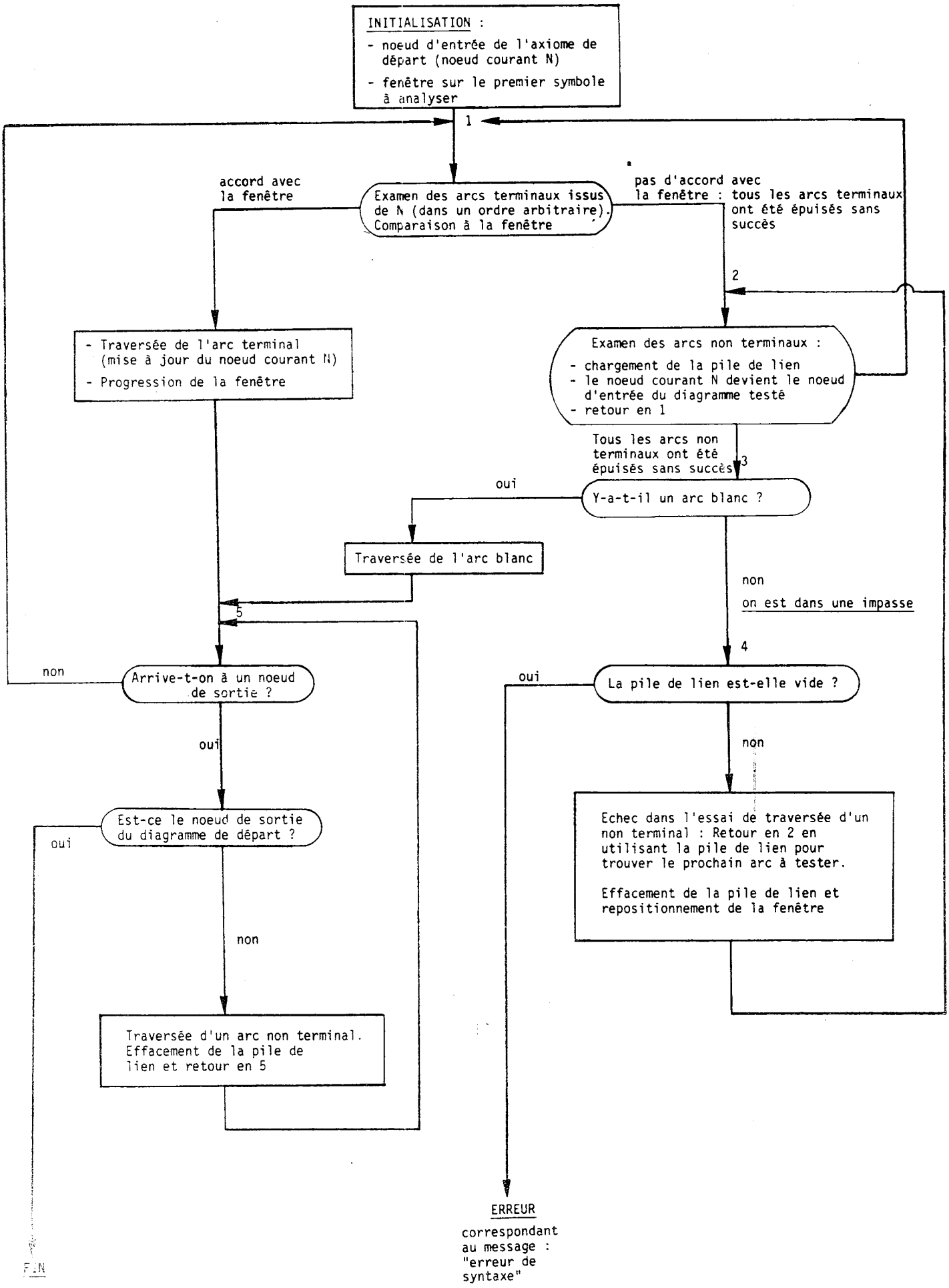
- Sinon, on se trouve à un noeud autre que le noeud de sortie du diagramme de départ, donc la phrase est fausse.

Ainsi, une condition nécessaire et suffisante pour qu'une phrase soit correcte est que simultanément :

- la fenêtre pointe sur le délimiteur de fin de chaîne ;
- le balayage des diagrammes à partir de cette chaîne aboutit au noeud de sortie du diagramme de départ.

Voyons l'algorithme de balayage des diagrammes de transition.

FONCTIONNEMENT DU BALAYAGE DES DIAGRAMMES



REMARQUES :

- Une "pile de lieu" conserve la trace des différents diagrammes en cours de traversée. Le bas de cette pile concerne toujours un arc du diagramme de départ.

- Il apparaît que l'on teste les arcs divergeants d'un noeud dans un ordre arbitraire. Ceci impose que les diagrammes soient construits de telle sorte que, quels que soient le noeud où l'on se trouve et le symbole repéré par la fenêtre, il n'y ait qu'une manière de quitter ce noeud. Pour cela, les règles grammaticales du langage LACSYST ont été reécrites selon les conventions classiques qui permettent l'analyse déterministe descendante des chaînes d'entrée. Pour une définition plus formelle, nous renvoyons le lecteur à l'un des nombreux ouvrages de théorie des langages (réf. GRIFFITHS [16] p. 23).

4.e.3. Implémentation Socrate de l'analyseur "batch"

Pour réaliser ce traitement par lots, nous devons :

a) Stocker cette grammaire LACSYST en mémoire.

b) Définir des programmes standards d'analyse et d'exécution des requêtes formulées en LACSYST.

Remarquons que :

- Les programmes d'analyse syntaxique sont indépendants de la grammaire qui doit diriger directement cette analyse.

- Il est évident que nous avons dû écrire des programmes d'analyse sémantique et d'exécution propres à chaque règle LACSYST, mais comme au § 4.a.1. nous avons défini des utilitaires équivalents (BINIT, BAMAJ, BACRE, BAMOD, ...) pour permettre une programmation structurée.

4.e.3.1. Mémorisation de la grammaire LACSYST

Définissons maintenant les principes de stockage des diagrammes de transition correspondant aux règles du langage de description : à titre d'exemple, nous utiliserons la règle de description de modules = <RMAJMO> (cf. p. 1.79).

Nous avons reécrit cette règle dans une notation sous forme de Backus :

<RMAJMO> ::= majmo <CODEMO> <TXTTN> <ENTRE1><S1>

Ici, nous appelons PRODUCTION les éléments figurant en partie droite d'une REGLE. Nous dirons que le vocable majmo est une "entrée" de la règle <RMAJMO> et les autres productions seront appelées éléments de type "suite".

Décrivons la règle <ENTRE1> :

<ENTRE1> ::= entrée <E1> suivant <S2>

<E1> ::= a <CODEAL><EA>|c <CODECO><EC>|m <CODEMO><EM>

Soit :

REGLE	PRODUCTIONS	TYPE DE LA PRODUCTION
E1	a	entrée
	CODEAC	suite
	EA	suite
	c	entrée
	CODECO	suite
	EC	suite
	m	entrée
	CODEMO	suite
	EM	suite

Chaque production est identifiée par un VOCABLE (les vocables sont les mots dans la colonne PRODUCTION). Une production peut être elle-même :

- une règle (exemple : EA, EC)
- un mot-clé (exemple : a, c, m)
- une valeur terminale (exemple : CODEAC, CODECO).

Enfin, lorsqu'il s'agit d'une valeur terminale, celle-ci peut être :

- un code
- un mot
- une ligne
- un texte

ou - une valeur numérique.

Dans l'exemple précédent, les productions CODEAC, CODECO et CODEMO ont une valeur terminale de type code.

Tous ces renseignements sur les règles sont rangés dans les entités REGLE et VOCAB de la structure Socrate de MACSI-1.

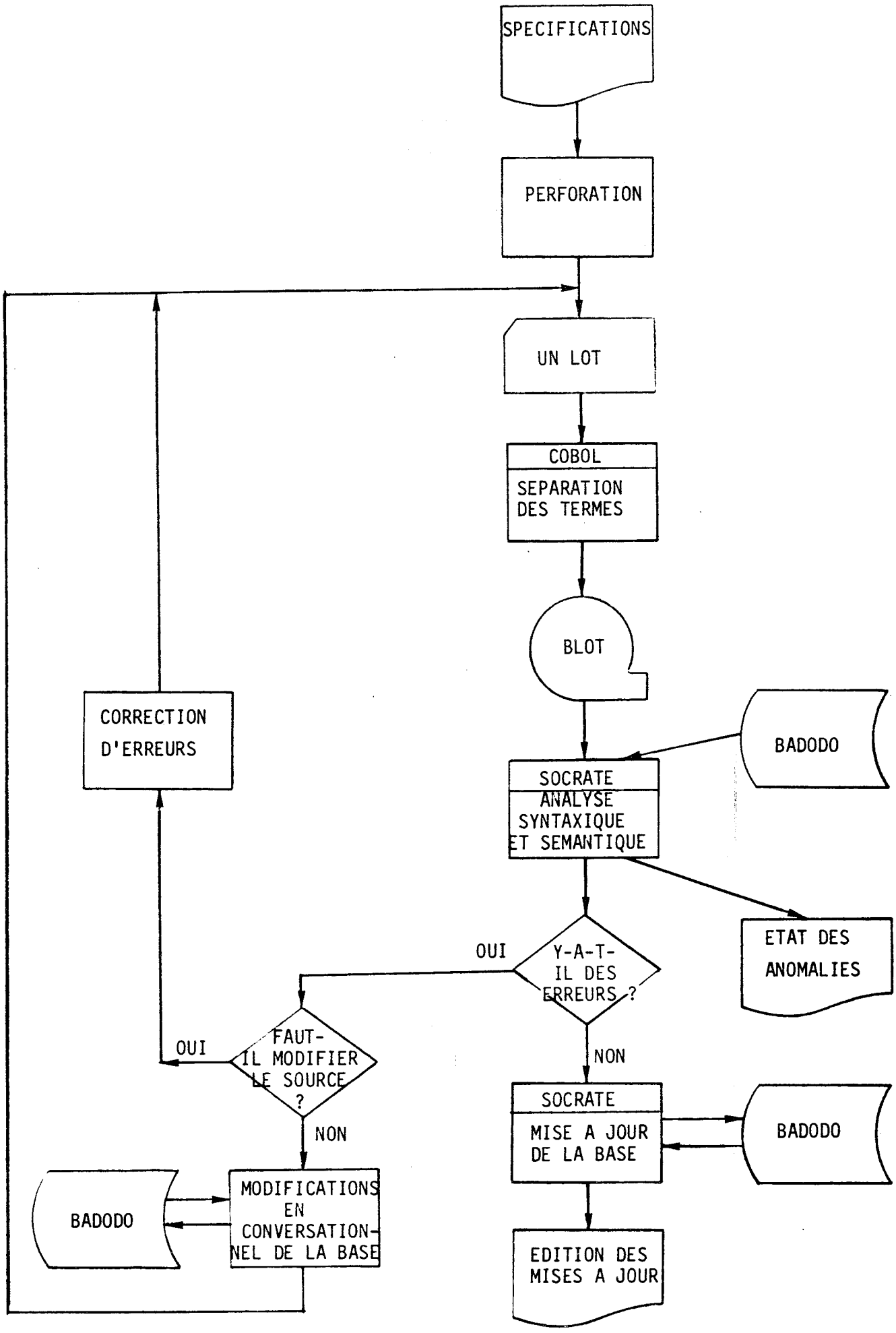
```
Entité 1000 REGLE /* Liste des règles de MACSI-1 */
Début
    NOM /* C'est le nom de la règle. Par exemple : RMAJMO, ENTRE1, E1 */
    RERREUR  /* C'est le numéro du message au cas où l'analyseur détecte
                une erreur */
    ENTREE   /* Liste de toutes les productions de type ENTREE dans la règle */
    Entité 100 PRODUCTION /* Liste des productions d'une règle */
    Début
        TERME  /* Vocable ou nom de la production */
        SUITE  /* Chaînage avec la production suivant */
        PERREUR /* Numéro du message au cas où l'analyseur détecterait
                   une erreur sur la production */
    fin
fin

Entité 1000 VOCAB /* Liste des vocables */
Début
    NOM /* Nom ou terme du vocable */
    TYPE /* 1 = règle vraie
            2 = règle regroupement
            3 = mot-clé
            4 = valeur terminale */
    DEFINITION /* Définition (au sens MACSI-1) du vocable */
    CONTROL-SYNTAX /* Type de contrôle sur ce vocable selon sa nature */
    CONTROL-SEMANT /* Nom du programme de contrôle sémantique sur ce vocable */
    EXECUT /* Nom du programme qui fait les traitements correspondants dans
              la base pour le vocable */
fin
```

4.e.3.2. Réalisation du traitement par lots

Un traitement par lots permet à l'analyste d'écrire ses spécifications sur papier libre en suivant la syntaxe des règles de description. Ces spécifications sont, ensuite, perforées et rentrées en machine pour être contrôlées puis par la suite mettre à jour la base de données documentaire.

Le processus suivant peut définir une entrée batch des spécifications :



4.e.3.2.1. Préparation du source

L'entrée batch est caractérisée par l'écriture des spécifications sur papier en format libre, mais en suivant les règles syntaxiques imposées par les règles de description. Le paquet de feuilles correspondant à une entrée batch est appelé un LOT.

Une opération de perforation recopie les spécifications du lot sur cartes perforées.

Le paquet de cartes est traité par un programme Cobol appelé "TXTINT". Ce programme décompose le texte source en isolant les mots qui sont séparés par des blancs dans le texte source. Avec une exception, néanmoins, lorsqu'il s'agit des textes (encadrés par deux étoiles). En effet, un texte sera décomposé en fractions de 30 caractères.

Le résultat de cette décomposition est enregistré sur une bande de travail (BLOT).

Le programme Cobol se charge aussi du tri des enregistrements sur la bande de façon à faire passer en tête les phases de création et de déclaration pour se donner un maximum de chance d'avoir tout déclaré avant de faire des mises à jour ou modifications. L'ordre de tri choisi est le suivant :

01 CRESE
02 CREPE
11 CREENS
12 CRERL
21 MAJMO
22 MAJCO
23 MAJAC
24 MAJGR
31 MAJDO
32 MAJPP
33 MAJPE
41 CREFI
51 MAJEGDO
61 LANCE

62 CREOP
63 ANALYSE
64 PROG
71 MAJFI
81 MAJEG
91 CREST
92 BONST
....

4.e.3.2.2. Analyse syntaxique et sémantique

Au § 4.e.2., nous avons vu qu'à partir de notre grammaire LACSYST il était possible de construire un analyseur déterministe descendant. Notre préoccupation ici n'est pas de redéfinir des concepts d'analyse syntaxique et sémantique, mais nous pouvons dire maintenant (avec GRIFFITHS réf. [16]) qu'il est possible d'inclure des fonctions sémantiques à n'importe quel point de notre grammaire car LACSYST est au moins de type LL(1).

Présentons de façon externe le principe d'introduction de ces fonctions sémantiques en rappelant l'exemple de <RMAJMO> évoqué au § 4.e.3.1..

②	③	①	④	⑤	⑥	⑦
REGLE		RMAJMO				
ENTREE	M	majmo				BMAJMO
SUITE	V	CODEMO	800	CODE	SECDM1	
SUITE	R	TXTTN	801			
SUITE	R	ENTRE1	802			
SUITE	R	S1	803			
⋮	⋮	⋮	⋮	⋮	⋮	
REGLE		ENTRE1	804			
ENTREE	M	entrée				
SUITE	R	E1	805			
SUITE	M	suivant				
SUITE	R	S2	807			
REGLE		E1	808			
ENTREE	M	a				
SUITE	V	CODEAC	809	CODE	SECDACE	
SUITE	R	EA	810			
ENTREE	M	c				
SUITE	V	CODECO	811	CODE	SECDCOE	
SUITE	R	EC	812			
ENTREE	M	m				
SUITE	V	CODEMO	813	CODE	SECDMOE	
SUITE	R	EM	814			
⋮	⋮	⋮	⋮	⋮	⋮	

TABLEAU 4

Colonne ① : Dans colonne ① sont groupées les règles et les productions nommées par leur vocable.

Colonne ② : Nous indique pour une règle donnée (REGLE) quelles sont ses productions. Les productions d'entrées sont marquées par "ENTREE" et leurs suivants par "SUITE".

Colonne ③ : Nous indique le type d'une production. Rappelons-nous qu'une production peut avoir l'un des types suivants :

R → Règle
M → Mot-clé
V → Valeur terminale

Colonne ④ : Nous sert à associer un numéro d'erreur à une règle ou à une production donnée. Le numéro correspond au code du message à éditer si l'analyseur détecte une erreur sur la règle ou sur la production référencée.

Colonne ⑤ : Correspond au contrôle à exécuter lorsqu'une production de type valeur terminale est rencontrée. Rappelons qu'une production de type valeur terminale peut être :

- un code
- un mot
- une ligne
- un texte
- une valeur numérique.

Pour un code, on vérifiera que c'est une suite de 8 caractères au maximum qui commence par un caractère alphabétique.

Un mot doit être une suite de 30 caractères au maximum (le blanc non compris).

Une ligne est une combinaison de 60 caractères au maximum (y compris les blancs).

Un texte est une suite de 100 lignes au maximum.

Une valeur numérique est une suite de 9 chiffres au maximum positive ou négative.

Colonne ⑥ : Indique le nom du programme qui exécute le contrôle sémantique sur une production de type valeur terminale. Généralement, les contrôles sémantiques se font sur les codes, c'est-à-dire qu'on vérifie la compatibilité entre l'opération qu'on veut faire et l'entité touchée. Par exemple, si on fait une opération de création d'une personne, il faudra vérifier

que le code que l'on donne pour cette personne est compatible avec cette création, c'est-à-dire qu'il n'y a pas déjà un DICO généré pour cette personne. En effet, plusieurs situations peuvent se présenter :

1) Le code correspond à la création d'un élément :

   a - On contrôle que l'élément n'existe pas : il n'y a pas un DICO avec ce code
   b - On génère le DICO correspondant et on indique que l'élément a été créé (TOPVAL = 6).

2) Le code correspond à une mise à jour d'un élément :

   a - On contrôle que l'élément existe : il y a un DICO avec ce code
   b - On contrôle que l'élément est disponible pour une mise à jour, c'est-à-dire que TOPVAL = 2
   c - On indique que l'élément a été mise à jour (TOPVAL = 6).

3) Le code correspond à une modification d'un élément (NOTE : les modifications en batch sont très rares, seulement la modification de modules modmo a été implémentée) :

   a - On contrôle que l'élément existe : il y a un DICO avec ce code
   b - On contrôle que l'élément est disponible pour une modification, c'est-à-dire que le TOPVAL = 4 ou 6
   c - On indique que l'élément peut être éventuellement modifié par la suite (TOPVAL = 6).

4) Le code correspond à une citation d'élément lors d'une création, mise à jour ou modification. Deux cas peuvent se présenter :

   I) L'élément est cité pour la première fois (déclaration), c'est-à-dire qu'il n'existe pas dans le DICO :
      I-a) Un DICO sera généré pour l'élément
      I-b) On indique que c'est le libellé de l'élément qui doit suivre
      I-c) On indique que l'élément est disponible pour une mise à jour (TOPVAL = 2).

   II) L'élément existe déjà dans la base : il y a un DICO avec ce code :

II-a) On indique que le libellé de l'élément ne doit pas suivre dans les spécifications.

Pour faire tous ces tests et garder les indications, nous nous servons de plusieurs éléments :

1) Du DICO et ces éléments en particulier :

- CODE
- TOPVAL
- INITIALISATEUR

2) Du TOPVAL des éléments

3) Des chaînes d'inverse dans la structure qui nous permettent d'enregistrer les éléments du DICO qui ont été manipulés dans une session :

DICO-TOU /* Eléments du DICO TOUchés */
DICO-NOU /* NOUveaux éléments dans le DICO */
DICO-CRE /* Eléments CREés */
DICO-MAJ /* Eléments Mis A Jour */
DICO-MOD /* Eléments MODifiés */

D'autres contrôles sémantiques sont envisagés :

- Vérification qu'une valeur numérique est comprise dans une plage de valeurs
- Contrôle sur le nombre de réalisations d'une entité, par exemple on peut vérifier qu'une personne n'a pas plus de 5 rôles
- etc ...

Colonne (7) : Indique le nom du programme à exécuter pour mettre à jour la base de données si les contrôles syntaxiques et sémantiques n'ont signalé aucune erreur.

4.e.3.2.3. Mise à jour de la base

A l'issue du passage sans erreurs d'un lot en analyse syntaxique et sémantique, nous disposons alors d'un texte source représentant une suite cohérence de phrases en "LACSYST".

un LOT

creens E1
...
fcreens

une phrase LACSYST

majmo M1
suivant c C10 ...
suivant a A17 ...
fmajmo

majco C10
...
fmajco

majac A17
...
fmajac

majdo D1
...
fmajdo

fin du lot — *FIN*

Il faut faire exécuter réellement ces requêtes LACSYST. L'exécution de chaque requête doit être dirigée par un programme d'exécution standard que nous avons alors introduit au niveau de la grammaire (une fonction d'exécution est analogue à une fonction sémantique particulière).

Dans le § 4.e.3.2.2., le tableau nous précise que nous indiquons le nom du programme d'exécution à activer en colonne 7 et l'associons au mot-clé (majmo, majco, majac ...) caractérisant une requête.

Le déroulement de cette "passe de mise à jour" est alors très simple si on sait que les mots-clefs majmo, majco, majac, ... sont les entrées des règles directement dérivées de l'axiome de la grammaire. Décrivons une telle mise à jour à partir du lot précédent.

a) - La lecture du mot-clé majmo nous permet d'identifier le programme d'exécution associé au vocable majmo.

b) - L'activation du programme "bmajmo" permet l'exécution de toute la requête majmo M1 ..... fmajmo .

c) - On refait a) sauf si on est à la "FIN" du lot.

Toute requête LACSYST est donc traitée par un programme d'exécution standard (bmajmo, bmajco, bmajac, ...). Comme pour la programmation des opérateurs du mode conversationnel (§ 4.d.), nous avons défini des programmes utilitaires pour faire une programmation structurée et souple de ces opérateurs utilisables dans le traitement par lots. Ces utilitaires ont en général les mêmes fonctions que leurs homologues conversationnels (on aura : BINIT, BAMAJ, BAMOD, BACRE, BERREUR, ...).

4.f. Quelques extensions

Nous allons présenter, par un exemple concret, les possibilités d'extension de MACSI-1.

4.f.1. Le composant RECLAMATION

En effet, une extension de MACSI-1 a été effectuée d'après le souhait d'un utilisateur qui voulait voir l'outil lui permettre de faire une gestion automatique des textes exprimant toutes les difficultés rencontrées par les utilisateurs de MACSI-1 pour un projet donné.

Il fallait ajouter un composant RECLAMATION (nous n'avons pas eu de difficulté car Socrate nous permet une définition dynamique des entités) et les commandes de gestion de ces Réclamations (création, modifications, suppression) dont l'"implémentation n'a pas posé des problèmes particuliers.

Nous décrivons fonctionnellement ce composant Réclamation par le tableau suivant :

COMPOSANT = RECLAMATION

PROPRIETES		RELATION AVEC D'AUTRES COMPOSANTS			PROPRIETES DE CES RELATIONS	
Identificateur	Explication	Identificateur	Composant en relation	Explication	Identificateur	Explication
CODERE	Code d'identification de la réclamation					
DEFRE	Définition de la réclamation					
ETATRE	Etat de la réclamation					
DATE-1	Date de formulation de la réclamation					
DATE-2	Date de prise en compte de la réclamation					
DATE-3	Date de réalisation de la réclamation					
DATE-4	Date de refus de la réclamation					
		AUTEUR-RE	PERSONNES	Auteur de la réclamation		
		EXAMINATEUR	PERSONNES	Examinateur de la réclamation		

REMARQUES :

(1) Le composant RECLAMATIONS nous permet d'enregistrer et par la suite d'étudier les questions, remarques, souhaits des utilisateurs par rapport au produit MACSI-1 ou par rapport à l'application en cours de spécification.

(2) Les différents états (ETATRE) possibles d'une Réclamation sont les suivants :

1 - Réclamation enregistrée en attente d'analyse

2 - Réclamation en cours d'examen

3 - Réclamation prise en compte et réalisée

4 - Réclamation refutée après étude.

Il faut alors définir une commande de création d'une réclamation :

crere CODERE * DEFINITION * fcrere

De même, les programmes de gestion d'une réclamation devront être spécifiés.

4.f.2. Les opérateurs

Une autre extension possible porterait sur les opérateurs. En effet, nous avons donné pour un opérateur le schéma de structure suivante :

```
Entité OPERATEUR
   début
        CODE
        LIBELLE
        SPECIF
        SOURCE
        :
        :
   fin
```

(Le lecteur trouvera la structure complète en ANNEXE A).

La caractéristique SPECIF nous permet de donner une définition qualitative de l'opérateur, mais aussi d'enregistrer sa description logique sous forme de schémas de programmes. Le SOURCE nous permet de garder une copie du programme correspondant à l'opérateur.

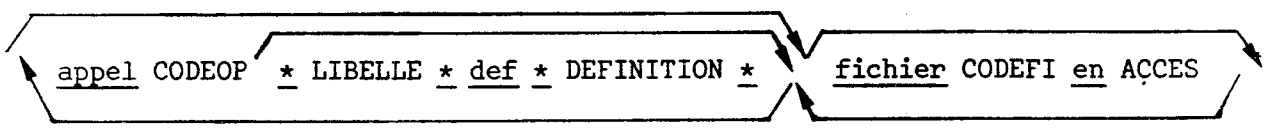
D'après la présentation que nous avons fait des "programmes de service" (cf. § 4.d.1.), il nous semble qu'une structure plus complète de représentation des opérateurs peut être envisagée :

```
Entité OPERATEUR
   début
        CODE
        LIBELLE
        OBJET
        SYNTAXE
        SCHEMA-DE-PROGRAMME
        EXEMPLE-D'UTILISATION
        SOURCE
        :
   fin
```

où l'on remplace avantageusement la caractéristique SPECIF par les 4 caractéristiques : OBJET, SYNTAXE, SCHEMA-DE-PROGRAMME et EXEMPLE-D'UTILISATION.

La commande ANALYSE (cf. 2.26) serait alors modifiée pour accepter ces nouvelles spécifications.

analyse CODEOP type TYPE accès ACCES mode MODE



objet * OBJET * syntaxe * SYNTAXE * schéma * SCHEMA-DE-PROGRAMME *

ex * EXEMPLE * fanalyse

4.g. Programmes de documentation et de contrôle

L'un des objectifs prioritaires de MACSI-1 est de contrôler la cohérence des spécifications produites et de faciliter leur diffusion (cf. § 1.b.4.). Certains principes de contrôle ont déjà été évoqués au cours de ce travail. Cependant, nous devons faire une récapitulation des contrôles et des éditions fournies par MACSI-1.

En effet, on peut distinguer deux types d'opérations fondamentales sur les spécifications :

- les contrôles,
- la documentation.

Nous entendons par contrôle les opérations manuelles ou automatiques effectuées sur la base documentaire pour corriger des anomalies constatées ou bien pour améliorer les spécifications. Bien entendu, de nombreuses éditions seront faites par les programmes de PACS pour les responsables du projet et des analystes afin de leur permettre de réaliser des vérifications. Les programmes de PACS assurent aussi la documentation totale ou partielle du projet à partir de la base documentaire. Ces éditions ont un grand intérêt car elles permettront de faire le point sur l'état de la base documentaire.

Nous ne pourrons pas donner ici une liste complète de tous les contrôles et de toutes les éditions (liées aux contrôles ou liées à la documentation), car ils sont trop nombreux et cette liste n'est pas limitée. En effet, on pourrait toujours imaginer de nouveaux contrôles et éditions possibles. Néanmoins, chaque projet étant différent, seul un sous-ensemble de programmes de contrôle et éditions possibles sera à prendre en compte. PACS sera donc constituée par un noyau de programmes standards, celui-ci pouvant être enrichi par des programmes particuliers à chaque application et demandes "à la carte" par les utilisateurs de MACSI-1 avant, pendant ou après l'analyse.

1) Les programmes de contrôle

Avant de définir les différents types de programmes de contrôle, il faut souligner que de nombreux contrôles dans MACSI-1 sont implicites, c'est-à-dire qu'ils sont inhérents aux règles de description et aux contraintes du modèle. Nous l'avons vu au cours de ce travail, néanmoins rappelons-en quelques-uns :

- Les spécifications des composants et des données de ces composants sont introduites selon un ordre logique strict.

- Vérification des droits de déclarer, créer, compléter, modifier ou supprimer un élément d'un composant.

- Obligation de "valoriser" les propriétés des composants.

- etc ....

Cependant, d'autres programmes de contrôle sont à prendre en compte. On peut distinguer deux catégories :

a) Les programmes qui peuvent, à un certain moment de l'analyse et selon l'état de la base documentaire, bloquer éventuellement l'avancement de celle-ci :

   . Interdire le passage aux spécifications organiques tant que les spécifications fonctionnelles ne sont pas terminées.

   . Vérification que dans une décomposition de propriétés composées, on ne produit aucun cycle.

   . Vérification que, pour toutes les boucles dans un module, il y a toujours un élément qui permet la "sortie" de la boucle.

   . Interdire les appels successifs de modules.

   . etc ....

b) Les programmes de diagnostic et bilans sur les spécifications produites. Ces programmes ont pour tâche d'éditer les spécifications de façon à montrer de possibles anomalies dans l'état actuel de la base documentaire ou bien de faire ressortir certaines relations, entre les composants du projet, jugées comme d'intérêt pour le contrôle des spécifications. Certains de ces programmes seront activés automatiquement, d'autres seront activés à la demande des utilisateurs.

Voyons quelles seraient les principales éditions associées à ces programmes de diagnostics et de bilans :

- Inventaire des spécifications incomplètes, c'est-à-dire édition de la liste par composant (et pour tous les composants éventuellement) des éléments ayant des caractéristiques avec la valeur "u".

- Edition des éléments déclarés mais non définis.

- Edition des propriétés non enregistrées dans la structure Socrate associée au projet.

- Edition des "cross-references" entre les composants.

- Edition des éléments en état "privé".

- Liste des personnes dans un groupe qui sont rattachées à différents services administratifs.

- Liste des personnes non rattachées à un groupe.

- Liste des personnes appartenant à différents groupes d'exécution.

- Edition des groupes d'exécution n'ayant que des commandes de sortie.

- Edition des groupes d'exécution n'ayant que des commantes d'entrée.

- Edition des groupes d'exécution ayant les mêmes actions ou les mêmes commandes associées.

- Liste des actions, commandes ou modules qui apparaissent comme éléments dans plusieurs modules.

- Edition des cycles dans les modules.

- Edition des commandes sans documents associés.

- Edition des commandes à implémenter.

- Edition des commandes à l'état 2.

- Edition des commandes à l'état 3.

- Liste des opérateurs de service.

- Liste des opérateurs non réceptionnés après la date prévue.

- Edition des explications avec la valeur "ras".

- Edition des réclamations.

- etc ....

2) Les programmes assurant la documentation

Ces programmes doivent assurer l'édition de tout ou partie de la base documentaire de façon à produire ce qu'on appelle communément les "dossiers d'analyse". Néanmoins, à certains moments de l'analyse, des éditions de caractère général seront produites à l'intention de tous les utilisateurs. Des programmes d'édition de glossaires et des états récapitulatifs ou statistiques seront aussi implémentés. Ici encore, certains de ces programmes seront activés automatiquement et d'autres à la demande.

Donnons quelques-uns de ces programmes dans leur ordre d'activation au cours de l'analyse :

. Dans une commande MACSI-1 : par exemple les programmes d'édition pour la récapitulation d'une opération (cf. EDICO p. 3.8).

. A la suite d'une commande de MACSI-1 :
  - éditions des caractéristiques d'un composant lorsqu'il passe à l'état public (TOPVAL = 8) (cf. p. 4.9),
  - édition des éléments déclarés par une création, mise à jour ou modification d'un élément (cf. p. 4.34).

. Périodiquement = édition d'un bilan provisoire (cf. BILPROV p. 2.12) pour informer les analystes de l'état des spécifications et de l'avancement du projet.

. Entre les étapes de l'analyse :
  - à la fin de la description logique des données (cf. BILDONN p. 2.4)
  - à la fin de la description de l'organisation (cf. BILORGA p. 2.4)
  - au passage de l'analyse fonctionnelle à l'analyse organique et à la programmation (cf. EDITFON p. 2.4).

Naturellement, il est possible de définir tout programme de documentation sélective explicitement demandé par les utilisateurs. Il est bien évident que nous ne pouvons pas proposer avec PACS une liste exhaustive de ces programmes.

Donnons-en quelques exemples :

- Un chef de service demandera :

  Pour un service donné, liste des personnes y appartenant avec, pour chaque personne, liste des actions dont elle est responsable et groupe d'exécution associé à chaque action.

- Le service tirage demandera :

  Liste de documents avec sa définition et description complète des propriétés associées.

- Le responsable du service informatique demandera :

  Liste de toutes les personnes qui sont responsables du lancement d'une ou plusieurs commandes avec leur service administratif associé.

- Le service d'exploitation aimerait connaître la liste des fichiers et les supports correspondants prévus pour cette application.

- Le chef du personnel demandera :

  Liste des personnes avec leur fonction
  Liste des personnes ayant 5 rôles
  Liste des personnes sans rôle.

- Le chef du projet demandera :

  Contenu d'un ou plusieurs composants
  Edition graphique des modules
  Arborescence des opérateurs
  Edition des composants classés selon le TOPVAL.

- etc, etc ....

CHAPITRE 5

CONCLUSIONS

Au terme de la présentation de ces résultats techniques, nous nous permettons d'esquisser auprès du lecteur les développements que nous en attendons au niveau de la pédagogie, développements d'autant plus souhaités qu'à l'issue de ce travail nous aurons à assurer des responsabilités d'enseignement de l'analyse en informatique de gestion à l'I.U.T. de CARACAS.

Dans cette perspective, nous avons participé à l'enseignement de l'analyse à l'I.U.T.-B de Grenoble, nous avons pris connaissance de certains programmes qui, dans cette discipline, ont été mis en place dans d'autres I.U.T. et, enfin, comme le lecteur a pu le remarquer, nous avons étudié les tendances qui se font jour récemment dans le cadre de l'enseignement de la programmation (réf. [5], [8], [12]).

Quelles sont les remarques fondamentales que nous a suggéré l'observation de ces activités pédagogiques et, à partir de celles-ci, quels sont les progrès que nous souhaiterions personnellement dans l'enseignement de l'analyse ?

L'enseignement de la programmation nous a beaucoup fait réfléchir quant à l'évolution qu'il a subie ces dernières années. Initialement, considéré comme l'apprentissage d'un langage (Cobol, Fortran, PL/1 ou autres), cet enseignement dégagea progressivement des schémas généraux de construction de programmes, les langages de programmation n'étant alors considérés que comme des moyens de traduction de ces schémas généraux. Par ailleurs, il a été fait de plus en plus usage de moyens conversationnels pour familiariser les étudiants avec l'algorithmique au niveau de l'initiation (utilisation d'APL en particulier).

Parallèlement dans le cadre de l'enseignement de l'analyse, nous savons que certains établissements ont associé leur enseignement à une méthode d'analyse particulière (Warnier, Corig notamment) ; d'autres ont renoncé à enseigner une méthode bien spécifique. Quelle que soit la décision de prendre, ou non, une méthodologie particulière,

l'enseignement de l'analyse nous donne, aujourd'hui, l'impression désagréable de fournir aux étudiants plus un ensemble de recettes qu'une formulation précise,structurée et générale des mécanismes fondamentaux de conception et de construction d'un projet informatique.

Nous voudrions alors expliquer, très sommairement, dans quelle mesure les travaux que nous présentons ici nous laissent espérer un progrès dans la structuration de l'enseignement de l'analyse selon une évolution comparable à celle qui a fait ses preuves dans l'enseignement de la programmation.

1) Possibilités pédagogiques apportées par les caractéristiques techniques de MACSI-1

a) Exprimer par des concepts clairs les spécifications d'un projet d'abord au niveau de l'analyse fonctionnelle puis de l'analyse organique :

De même que nous avons décrit aux chapitres 1 et 2 les caractéristiques fonctionnelles d'un projet et au chapitre 3 ses caractéristiques organiques, de même nous pensons pouvoir, par une présentation appropriée de ces concepts, offrir aux étudiants un schéma de référence précisant ce qu'il faut entendre par analyse fonctionnelle et analyse organique. Pour eux, actuellement la distinction est relativement floue : l'enseignement de ce qui est appelé "analyse fonctionnelle" n'est pas étayé par des notions précises descriptives de la nature d'une organisation administrative ou d'entreprise, des traitements automatisés et de l'interface entre les deux. Il nous semble que les notions de module, de structure logique de données, d'action, de commande permettent de fournir un autre discours plus opérationnel pour les étudiants.

b) Initiation conversationnelle puis utilisation en "batch" :

Tout comme il nous paraît extrêmement efficace de faire une initiation à l'algorithmique pour des étudiants à partir de terminaux, en utilisant, en mode conversationnel, un langage comme Fortran, APL ou Basic, de même il nous paraîtrait extrêmement productif de faire un apprentissage du langage de description LACSYST sous une forme conver-

sationnelle. Nous imaginons, en effet, très bien que des étudiants puissent commencer la description d'un petit projet sous un mode conversationnel dans lequel ils seront immédiatement sanctionnés tant au niveau de l'incorrection syntaxique de leurs descriptions qu'au niveau des insuffisances mises en évidence par certains programmes de contrôle sémantique. Lorsqu'un certain nombre de mécanismes seront ainsi rapidement acquis, nous pourrons demander dans le cadre de la description d'un projet, l'utilisation du langage LACSYST sous une forme de traitements par lots en temps différé.

c) Synthèse possible entre certains mécanismes de construction de programmes et des mécanismes de construction des projets :

Le lecteur aura certainement remarqué qu'au niveau de la construction de modules, nous avons fait appel à des mécanismes d'analyse descendante : ils ne nous sont pas propres et nous avons volontiers avoué les avoir réinterprétés à partir des publications qui les présentaient au niveau de la programmation. Néanmoins, ils nous semblent, à l'expérience, adéquats non seulement pour construire des programmes mais aussi pour construire des projets et, peut-être, plus généralement pour construire des systèmes. Il serait alors sans doute souhaitable, sinon possible, au niveau d'un enseignement de synthèse proposé à la fin d'une scolarité, de mettre en évidence auprès des étudiants cette similitude des mécanismes de construction qu'auraient séparément présentés, au préalable, un enseignement de programmation et un enseignement de MACSI-1. Tout effort de coordination pédagogique qui vise à contrebattre dans l'esprit des étudiants leur tendance naturelle à cloisonner les cours qui leur sont proposés en différentes spécialités qui s'ignorent, nous semble particulièrement nécessaire.

2) Esquisse d'une progression pédagogique de mise en oeuvre de MACSI-1

Quelles que soient les possibilités, que nous croyons réelles, de faire progresser l'enseignement de l'analyse par la mise en oeuvre de MACSI-1, il n'en reste pas moins à bâtir totalement une mise en oeuvre pédagogique efficace de cet outil. Ceci n'ira pas sans difficulté car nous n'avons encore que très peu d'expérience dans ce domaine.

Pour susciter un dialogue avec le lecteur, que nous espérons fructueux, nous voulous cependant livrer ici quelques suggestions qui, certes, donneront lieu à révision, mais qui représentent l'état actuel de nos réflexions sur une modalité concrète de mise en oeuvre de MACSI-1 dans un enseignement d'analyse.

L'objectif principal nous paraît aujourd'hui d'obtenir des étudiants qu'ils dépassent l'acquisition du formalisme associé à l'outil pour vraiment pouvoir utiliser cet outil dans l'analyse d'un projet. Or, une difficulté est évidente : toute tentative d'apprentissage de MACSI-1 serait un échec si les étudiants ne retenaient que les règles de description sans voir la finalité de ce qu'elles veulent décrire, s'ils ne s'intéressaient qu'aux mécanismes de description en oubliant de porter attention à l'objet décrit lui-même. Il faudra donc minimiser au maximum le temps d'apprentissage des règles de description et de la structure du modèle qui leur est associé pour exiger, le plus tôt possible, des étudiants la mise en oeuvre de ces règles dans la conception d'un projet informatique réel. Pour cela, une possibilité de progression nous paraît être la suivante :

1ère étape : Description extrêmement rapide des caractéristiques du modèle et des règles de description (8 heures).

2ème étape : Mise en oeuvre de ces règles de description par la présentation d'un projet complètement spécifié grâce à elles

Le projet décrit devra être choisi pour qu'il ne présente aucune difficulté au niveau de la compréhension de ses objectifs, de l'organisation administrative qu'il suppose et des traitements informatiques qu'il met en oeuvre. Il sera l'occasion d'apprendre, par une étude de cas,

l'utilisation réelle des règles de description et de donner aux étudiants l'occasion de prendre du recul par rapport à elles (20 heures).

3ème étape : Perfectionnement dans l'utilisation des règles par leur mise en oeuvre effective en mode conversationnel. Il sera probablement judicieux de demander aux étudiants d'utiliser ces règles non pas pour décrire un nouveau projet mais pour apporter des modifications, ou des complèments, au projet qu'on leur aura présenté dans l'étape précédente (4 séances de 1 heure par étudiant).

4ème étape : L'outil MACSI-1 étant considéré, à ce stade de la progression pédagogique, comme parfaitement connu de l'étudiant en ce qui concerne l'utilisation des règles de description, on lui demandera, à ce moment là, de s'en servir comme langage de description d'un nouveau projet, nouveau projet dans lequel il devra mettre en oeuvre essentiellement ses qualités d'analyse, d'imagination d'une solution et d'esprit critique vis à vis des incohérences du projet qu'il propose. Les spécifications qu'il fournira seront écrites sur des bordereaux, seront perforées, introduites en machine en "batch" et les programmes de contrôle de MACSI-1 lui fourniront un certain nombre de diagnostics.

5ème étape : A la fin de l'étude du projet, l'étudiant remettra non pas un dossier manuscrit décrivant son projet, mais fournira à l'enseignant d'analyse l'édition dans leur version définitive des spécifications telles qu'il les a rentrées dans la base documentaire.

Il est bien évident que la mise en oeuvre de MACSI-1 sous cette forme suppose des moyens de dactylocodage importants car le moindre petit projet, décrit avec les règles de description de LACSYST, suppose au minimum un volume de données de l'ordre de 2000 à 3000 cartes. Il faut donc que des possibilités importantes de perforation soient disponibles au niveau de l'établissement d'enseignement pour que la progression pédagogique présentée ci-dessus soit effectivement réalisable.

3) Une erreur à ne pas commettre

Nous avons souligné tout au long des pages précédentes notre préoccupation de donner à MACSI-1 une flexibilité qui permette de dégager une méthode de construction de méthode d'analyse. Pour cela, nous avons pris des choix d'implémentation permettant de modifier facilement tant la structure Socrate que les règles de description et les programmes associés. Cette flexibilité nous semblait indispensable dans la mesure où nous pensons qu'une méthode d'analyse n'est qu'un outil pour concevoir un projet et que, si possible, l'outil doit être adapté à ses conditions d'utilisation. Il est bien évident que cette préoccupation doit être aussi prioritaire dans l'enseignement de MACSI-1. Il nous semble nécessaire d'avoir constamment à l'esprit dans la conception d'un enseignement de l'analyse utilisant MACSI-1 l'erreur à ne pas commettre qui serait celle de terminer un enseignement de l'analyse utilisant MACSI-1 dans les conditions où nous venons de les présenter au paragraphe précédent. Il manque, en effet, dans la progression pédagogique que nous avons proposée, une dernière étape qui nous paraît essentielle : celle qui consisterait à demander aux étudiants une critique de la méthode qu'ils ont utilisée, suivie d'une autre étape leur demandant de modifier à la fois la méthode et les programmes qui lui sont associés. Si la possibilité de critique et de modification de MACSI-1 n'est pas fournie aux étudiants, ils sortiront de l'enseignement de l'analyse avec l'acquisition de ce qu'ils considéreront comme une méthode particulière, une méthode de plus ; ce n'est absolument pas ce que nous visons. Il nous paraît essentiel, en effet, qu'ils aient appris non pas une méthode, mais une technique pour se définir une méthode adaptée à leurs besoins.

La présentation technique des programmes d'aide à l'analyse associés à MACSI-1 est possible si les étudiants suivent, parallèlement à l'enseignement de l'analyse, un cours de Socrate (c'est le cas actuellement à l'I.U.T. de Grenoble). Si dans le cadre de cet enseignement sur les logiciels de base de données, la structure documentaire de MACSI-1 était présentée comme thème d'un travail pratique et que dans une autre séance de travail, il était demandé aux étudiants de modifier

tel ou tel programme de documentation ou bien s'il leur était présenté la structure générale de l'interpréteur que nous avons décrit (§ 4.c.1.), il nous semblerait alors possible, en synthèse de l'enseignement de l'analyse d'une part et de l'enseignement de bases de données d'autre part, de donner un exercice en fin de scolarité qui aurait pour objet d'une part la modification de la méthode MACSI-1 ou son extension, d'autre part les modifications correspondantes des programmes provoquées par cette modification.

Nous avons souvent rencontré des étudiants qui, après deux années de scolarité pendant lesquelles ils avaient appris la construction de schémas de programmes, leur traduction en Cobol, en Fortran, en Assembleur, craignaient néanmoins d'être incompétents parce qu'ils n'avaient pas fait en plus du PL/1, du GAP et d'autres langages. Malgré des efforts pédagogiques importants, ils avaient, à la veille de leur vie professionnelle, l'impression d'être prisonniers d'un outil. Peut-être qu'à l'issue de l'utilisation de MACSI-1, ils craindront aussi de na pas connaître les méthodes Warnier, Corig, Ariane, Minos, etc ... : nous souhaiterions au moins pouvoir leur répondre lorsqu'ils exprimeront ce genre d'inquiétude.

ANNEXE A

STRUCTURE SOCRATE DU PROJET MACSI-1

```
DEBUT
/* ------------------------------------------------------------ */
ENTITE 12 DICINIT
DEBUT
   PROJET DE 1 A 10
   VOTRE-CODE MOT 8
   DATE MOT 8
   OPERATION MOT 8
   CODE-EN-COURS MOT 8
   S-Y20 DE 0 A 1000000000
   Z16 MOT 30
   Z17 MOT 30
   EMETT MOT 8
   MESS1 MOT 30
   MESS2 MOT 30
   CODE MOT 12
   TOPVAL DE 0 A 1000000
   REPONSE MOT 1
   LIBELLE TEXTE 1
   LIB TEXTE 1
   NOM MOT 30
   PRENOM MOT 30
   ADRESSE TEXTE 1
   TELEPHONE MOT 30
   MEMBRE MOT 8
   EMPLOYE MOT 8
   DEFINITION REFERE UNE EXPLIC
   SOURCE REFERE UNE EXPLIC
   OPERATEUR MOT 8
   A-VOUS MOT 30
   FICHIER MOT 8
   ADMINISTRATION MOT 8
   TYPE MOT 4
   ETAT MOT 4
   NATURE MOT 4
   COMMANDE MOT 30
   ACCES MOT 6
   SI- REFERE UNE EXPLIC
   QUAND REFERE UNE EXPLIC
   ENTREE MOT 8
   PAR- MOT 8
   ELEM MOT 8
   SUIVANT- MOT 8
   NUMERO MOT 4
   MODIF-DE MOT 10
   MOT-CLE MOT 10
   LG-CODE MOT 1
   NIVEAU MOT 1
   DESIGNE TEXTE 1
   MODIFICATION REFERE UNE EXPLIC
   ACCORD MOT 1
   OK MOT 1
   EXPLICATIONS MOT 1
   SILENCIEUX MOT 1
   ENREGISTREMENT MOT 1
```

```
    TAILLE MOT 6
    ARG1 REFERE UN ENS-REL
    ARG2 REFERE UN ENS-REL
    MIN1 MOT 6
    MAX1 MOT 6
    MIN2 MOT 6
    MAX2 MOT 6
    PPCTRL REFERE UNE EXPLIC
    PTSE REFERE UN SERVADM
    PTPE REFERE UNE PERSONNE
    PTGR REFERE UN GROUFON
    PTMO REFERE UN MODULE
    PTAC REFERE UN ACTION
    PTCO REFERE UNE COMMANDE
    PTDO REFERE UN DOCUMENT
    PTEL REFERE UN ELEMENT DE UN MODULE
    PTDI REFERE UN DICO
    PTEX REFERE UNE EXPLIC
    PTFI REFERE UN FICHIER
    PTOP REFERE UN OPERATEUR
    PTRE REFERE UNE RECLAMATION
    PTE-R REFERE UN ENS-REL
    PTPP REFERE UNE PROPRIETE
    PTSS REFERE UNE STRSOCASS
    PTEG REFERE UN ENREG
    TPE INVERSE TOUT PERSONNE
    TGR INVERSE TOUT GROUFON
    TMO INVERSE TOUT MODULE
    TEL INVERSE TOUT ELEMENT DE UN MODULE
    TAC INVERSE TOUT ACTION
    TCO INVERSE TOUT COMMANDE
    TOP INVERSE TOUT OPERATEUR
    TDO INVERSE TOUT DOCUMENT
    TFI INVERSE TOUT FICHIER
    TE-R INVERSE TOUT ENS-REL
    TPP INVERSE TOUTE PROPRIETE
    TSS INVERSE TOUTE STRSOCASS
    TEG INVERSE TOUT ENREG
    TPC INVERSE TOUTE PROPRIETE
    ENTITE 100 CODCREE
    DEBUT
      CODE REFERE UN DICO
    FIN
    ENTITE 12 SAVEX      /* POUR MISE A JOUR DES EXPLICS */
    DEBUT
      EXP-ED REFERE UNE EXPLIC
      ENTITE 102 LIGNE
      DEBUT
        PTR-AR REFERE UNE LIGNE DE UN SAVEX
        LIB TEXTE 1
        PTR-AV REFERE UNE LIGNE DE UN SAVEX
      FIN
    FIN

FIN
```

```
ENTITE 10 PROJET
DEBUT
  NUMPROJ DE 1 A 10
  NUM-EX DE 0 A 15000
  NUM-RE DE 0 A 1500
  ENTITE 1000 CODE-A-LISTER
  DEBUT
    CODE MOT 8
  FIN
FIN

/* ------------------------------------------------------ */

/* ------------------------------------------------------ */

ENTITE 10000 EXPLIC
DEBUT
  CODE DE 1 A 15000 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  AUTEUR REFERE UNE PERSONNE
  SUPPRESSION DE 1 A 2  /*  1 NE PEUT ETRE SUPPRIMEE       *//
                        /* CAR CORRESPOND A UNE DEFINITION */
                        /*  2 PEUT ETRE SUPPRIMEE          */
  ENTITE 100 LISTE-DICO
  DEBUT
    VAL REFERE UN DICO
  FIN
  ENTITE 100 CONTENU
  DEBUT
    LIB TEXTE 1
  FIN
FIN

/* ------------------------------------------------------ */

ENTITE 3000 DICO
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  NUMERO DE 0 A 1000000
  TOPVAL DE 1 A 10
  INITIALISATEUR MOT 8
  NATURE DE 1 A 99
      /* SE GR PE MO AC CO DI OP ME FI DO EG ER PP SS */
  ENTITE 100 LISTE-EXPLIC
  DEBUT
    NUM-EDITION DE 1 A 100
    VAL REFERE UNE EXPLIC
  FIN
FIN

/* ------------------------------------------------------ */
```

```
ENTITE 1000 RECLAMATION
DEBUT
  CODE DE 1 A 1500 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  DEFINITION REFERE UNE EXPLIC
  AUTEUR REFERE UNE PERSONNE
  ETAT DE 1 A 99 /* 1=EN ATTENTE */
                 /* 2=EXAMEN     */
                 /* 3=EXECUTFE   */
                 /* 4=REALISEE   */
                 /* 5=REFUTEE    */
  DATE-1 MOT 8
  DATE-2 MOT 8
  DATE-3 MOT 8
  DATE-4 MOT 8
  DATE-5 MOT 8
  EXAMINATEUR REFERE UNE PERSONNE
FIN

/* ------------------------------------------------------ */

ENTITE 2000 MESSAGE
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  LIBELLE TEXTE 1
  DEFINITION REFERE UNE EXPLIC
  COMPTEUR DE 0 A 999999
  OPERATEUR-MES REFERE UN OPERATEUR
FIN

/* ------------------------------------------------------ */



/* ------------------------------------------------------ */

ENTITE 20 SERVADM
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  LIBELLE TEXTE 1
  DEFINITION REFERE UNE EXPLIC
  EMPLOYE INVERSE TOUTE PERSONNE
  ADRESSE TEXTE 1
  TELEPHONE MOT 10
FIN

/* ------------------------------------------------------ */
```

```
ENTITE 60 GROUFON
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10 /* 1 EN COURS D'INITIALISATION          */
                   /* 2 INITIALISE MAJ RESERVE A INIT.R   */
                   /* 3 MISE A JOUR EN COURS              */
                   /* 4 ERREUR RECONNUE PAR BILAN         */
                   /* 5 MODIFICATION EN COURS             */
                   /* 6 ACCES RESERVE A L'INITIALISATEUR  */
                   /* 7 ENTIEREMENT ACCESSIBLE            */
  LIBELLE TEXTE 1
  DEFINITION REFERE UNE EXPLIC
  MODINIT REFERE UN MODULE
  MOD-ASSOCIE INVERSE TOUT MODULE
  MEMBRE INVERSE TOUTE PERSONNE
  COMMANDE-UTILISABLE INVERSE TOUTE COMMANDE
  ACTION-ASSURE INVERSE TOUTE ACTION
FIN

/* ------------------------------------------------------- */

ENTITE  90 PERSONNE
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  NOM MOT 30
  PRENOM MOT 30
  ADMINISTRATION REFERE UN SERVADM
  ENTITE 5 ROLE
  DEBUT
    GROUPE REFERE UN GROUFON
    FONCTION REFERE UNE EXPLIC
  FIN
  ADRESSE TEXTE 1
  TELEPHONE MOT 10
FIN

/* ------------------------------------------------------- */

ENTITE 101 MODULE
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  LIBELLE TEXTE 1
  DEFINITION REFERE UNE EXPLIC
  MODINIT REFERE UN MODULE
  MOD-ASSOCIE INVERSE TOUT MODULE
  EL--ENTREE REFERE UN ELEMENT DE UN MODULE /* PT D'ENTREE */
  EL--SORTIE INVERSE TOUT ELEMENT DE UN MODULE
  ENTITE 90 ELEMENT
  DEBUT
```

```
      CODE MOT 8  /* CE CODE EST LE MEME QUE LE CODE MODULE */
                  /* ACTION OU COMMANDE QUI CONSTITUE       */
                  /* L'ELEMENT.                             */
      NATURE DE 4 A 6 /* CF NOMENCLATURE DES ENTITES */
      TOPVAL DE 1 A 10
      EXECUTANT REFERE UN GROUFCN
      PRECEDE INVERSE TOUT ELEMENT DE UN MODULE
      ENTITE 10 SUITE
      DEBUT
        ELEM-SUITE REFERE UN ELEMENT DE UN MODULE
        CONDITION-SUITE REFERE UNE EXPLIC
        QUAND REFERE UNE EXPLIC
      FIN
    FIN
  FIN

  /* ------------------------------------------------------ */

  ENTITE 200 ACTION
  DEBUT
    CODE MOT 8 AVEC CLE UNIQUE FIN
    PROJET DE 1 A 10
    TOPVAL DE 1 A 10
    LIBELLE TEXTE 1
    DEFINITION REFERE UNE EXPLIC
    MODINIT REFERE UN MODULE
    MOD-ASSOCIE INVERSE TOUT MODULE
    TYPE DE 1 A 99
    EXECUTANT INVERSE TOUT GROUFON
    ETAT DE 1 A 10   /* 1 FONCTIONNELLEMENT DEFINI          */
                     /* 2 MANUEL DEFINITION POSTE ECRIT     */
                     /* 3 SIMULATION FAITE                  */
                     /* 4 FONCTIONNE                        */
    RESULTAT REFERE UNE EXPLIC
  FIN

  /* ------------------------------------------------------ */

  ENTITE 500 COMMANDE
  DEBUT
    CODE MOT 8 AVEC CLE UNIQUE FIN
    PROJET DE 1 A 10
    TOPVAL DE 1 A 10
    LIBELLE TEXTE 1
    DEFINITION REFERE UNE EXPLIC
    MODINIT REFERE UN MODULE
    MOD-ASSOCIE INVERSE TOUT MODULE
    TYPE DE 1 A 99 /* 1=ENTREE  2=SORTIE */
    DOC-ASSOCIE INVERSE TOUT DOCUMENT
    EXECUTANT INVERSE TOUT GROUFON
    OPERATEUR-ASSOCIE REFERE UN OPERATEUR
    ETAT DE 1 A 10 /* 1=EN ATTENTE D'EXAMEN PAR EQUINF  */
                   /* 2=EN ATTENTE D'INFORMATION UTILIS */
                   /* 3=EN ATTENTE MODIF DE STUCTURE    */
                   /* 4=EN COURS DE REALISATION         */
```

```
                    /* 5=RECEPTION INFORMATIQUE              */
                    /* 6=RECEPTION UTILISATEUR               */
  DATE-1 MOT 8  /* DATE DE PRISE EN COMPTE          */
  DATE-2 MOT 8  /* DATE DE DEBUT ANALYSE            */
  DATE-3 MOT 8  /* DATE DE RECEPTION INFORMATIQUE */
  DATE-4 MOT 8  /* DATE DE RECEPTION UTILISATEUR  */
  ENTITE 50 MODIF
  DEBUT
    MODIFICATION REFERE UNE EXPLIC
    DATE-1 MOT 8  /* DATE DE PRISE EN COMPTE          */
    DATE-2 MOT 8  /* DATE DE DEBUT ANALYSE            */
    DATE-3 MOT 8  /* DATE DE RECEPTION INFORMATIQUE */
    DATE-4 MOT 8  /* DATE DE RECEPTION UTILISATEUR  */
    DATE-5 MOT 8  /* DATE D'ANNULATION                */
  FIN
FIN

/* ------------------------------------------------------------ */

/* ------------------------------------------------------------ */

ENTITE 1000 OPERATEUR
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  LIBELLE TEXTE 1
  TYPE DE  1 A 99
          /*  1 SAISIE          */
          /*  2 CONTROLE        */
          /*  4 TRAITEMENT      */
          /*  8 EDITION         */
          /* 16 CITATION        */
  USAGE DE 1 A 10
          /* 1 OPERATEUR PRINCIPAL     */
          /*  2 OPERATEUR SECONDAIRE   */
          /*  3 OPERATEUR DE SERVICE   */
  ETAT DE 1 A 10
          /* 1 EN COURS D'ANALYSE */
          /* 2 SPECIFIE */
          /* 3 PROGRAMME */
          /* 4 CONTROLE LIVRABLE */
          /* 5 LIVRE */
  NATURE DE 1 A 10
          /* 1 PROGRAMME */
          /* 2 MACRO-INSTRUCTION */
  ACCES DE 1 A 10
          /* 1 ACCEDE EN LECTURE A LA BASE */
          /* 2 ACCEDE EN ECRITURE A LA BASE */
          /* 3 ACCEDE EN LECTURE/ECRITURE */
          /* 4 N'ACCEDE PAS A LA BASE */
  MODE DE 1 A 10
          /* 1 CONVERSATIONNEL */
          /* 2 BATCH */
          /* 3 BATCH CONVERSATIONNEL */
```

```
    LANGAGE DE 1 A 10
            /* 1 SOCRATE */
            /* 2 IMT */
            /* 3 COBOL */
    SPECIF REFERE UN EXPLIC
    ESPDONNEE INVERSE TOUTE PROPRIETE
    SOURCE REFERE UN EXPLIC
    COM-ASSOCIE REFERE UNE COMMANDE
    OP-APPELLANT INVERSE TOUT OPERATEUR
    OP-APPELLE INVERSE TOUT OPERATEUR
    ENTITE 10 FICHIER-CONCERNE
      DEBUT
      ACCES DE 1 A 10
            /* 1 LECTURE */
            /* 2 ECRITURE */
            /* 3 LEC/ECR */
     REFICHIER REFERE UN FICHIER
      FIN
    ENTITE 50 HISTOIRE
    DEBUT
      MODIFICATION REFERE UNE EXPLIC
      REALISATEUR REFERE UNE PERSONNE
    FIN
FIN

/* ------------------------------------------------------- */

ENTITE 50 FICHIER
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  LIBELLE TEXTE 1
  TYPE DE 1 A 10
  DOC-ASSOCIE INVERSE TOUT DOCUMENT
  ORGANISATION DE 1 A 5    /* 1=SEQ 2=S.I. 3=DIRECT */
  SUPPORT DE 1 A 10 /* 1=BANDE  2=DISQUE 3=CARTE */
  VOLUME DE 0 A 100000000 /* NOMBRE DE CARACTERES */
  DESSIN-ENREG REFERE UNE EXPLIC
  CONTENU  REFERE UN ENREG
  KLE REFERE UNE PROPRIETE
  UTILISATION INVERSE TOUT OPERATEUR
            /* 1 FICHIER INTERNE DE TRAVAIL==>PAS DE DOCUMT */
            /* 2 FICHIER DE BASE ==> DOCUMENT ASSOCIES */
FIN

/* ------------------------------------------------------- */

ENTITE 500 DOCUMENT
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  LIBELLE TEXTE 1
  DEFINITION REFERE UNE EXPLIC
```

```
   CONTENU  REFERE UN ENREG
   KLE REFERE UNE PROPRIETE
   TYPE DE 1 A 99 /* 1=ENTREE 2=SORTIE */
   FICHIER-ASSOCIE INVERSE TOUT FICHIER
FIN

/* ------------------------------------------------------------ */

/* ------------------------------------------------------------ */

ENTITE 200 ENREG
DEBUT
   CODE MOT 8 AVEC CLE UNIQUE FIN
   PROJET DE 1 A 10
   TOPVAL DE 1 A 10
   LIBELLE TEXTE 1
   DESSIN REFERE UNE EXPLIC
   ESPDONNEE INVERSE TOUTE PROPRIETE
   FQUOI INVERSE TOUT FICHIER
   DQUOI INVERSE TOUT DOCUMENT
FIN

/* ------------------------------------------------------------ */

ENTITE 500 ENS-REL
DEBUT
   CODE MOT 8 AVEC CLE UNIQUE FIN
   PROJET DE 1 A 10
   TOPVAL DE 1 A 10
   TYPE DE 1 A 99      /* 1=ENSEMBLE, 2=RELATION              */
                       /*10=ENSEMBLE DE TRAVAIL               */
   LIBELLE TEXTE 1
   DEFINITION REFERE UNE EXPLIC
   TAILLE DE 1 A 99999     /* CARDINAL DE L'ENS OU REL        */
   PROPR INVERSE TOUTE PROPRIETE
   KLE INVERSE TOUTE PROPRIETE       /* INDEX,ARGUMENTS-TRIS*/
   ARG1 REFERE UN ENS-REL         /* 1ER ARGUMENT D'UNE REL  */
   ACCES1 MOT 30             /* NOM DE LA FONCTION D'ACCES   */
   MIN1 DE 0 A 99999
   MAX1 DE 0 A 99999
   ARG2 REFERE UN ENS-REL         /* 2EME ARGUMENT D'UNE REL */
   ACCES2 MOT 30
   MIN2 DE 0 A 99999
   MAX2 DE 0 A 99999
   PARTICIPE INVERSE TOUT ENS-REL  /* ENS-REL DE TYPE REL */
                     /* AYANT CET ENS-REL COMME ARGUMENT    */
FIN

/* ------------------------------------------------------------ */

ENTITE 1000 PROPRIETE
DEBUT
   CODE MOT 8 AVEC CLE UNIQUE FIN
   PROJET DE 1 A 10
   TOPVAL DE 1 A 10
```

```
  LIBELLE TEXTE 1
  DEFINITION REFERE UNE EXPLIC
  DEQUOI REFERE UN ENS-REL
  COMP-DE INVERSE TOUTE PROPRIETE    /* COMPOSANTE DE :... */
  TYPE DE 1 A 99      /* 1=COMPOSEE        5=NUM ENTIER      */
                      /* 2=TEXTE           6=NUM REEL        */
                      /* 3=MOT             7=LISTE ENTIERS   */
                      /* 4=LISTE MOTS      8=LISTE REELS     */
  PPCTRL REFERE UNE EXPLIC  /* EXISTENCE,COHERENCE....      */
  COMPOSANTS INVERSE TOUTE PROPRIETE     /* COMPOSEE DE:   */
FIN

/* ------------------------------------------------------------ */

ENTITE 5 STRSOCASS
DEBUT
  CODE MOT 8 AVEC CLE UNIQUE FIN
  PROJET DE 1 A 10
  TOPVAL DE 1 A 10
  LIBELLE TEXTE 1
  SOURCE REFERE UNE EXPLIC      /* STRUCTURE SOCRATE        */
  DATE-1 MOT 8          /* INITIALISATION-DEBUT ANALYSE     */
  DATE-2 MOT 8          /* RECEPTION INFORMATIQUE           */
  DATE-3 MOT 8          /* ANNULATION                  */
  ENTITE 1000 ELEMENT
  DEBUT
    CODE MOT 8
    TYPE DE 1 A 99    /* 1=ENS.,2=REL.,3=PROP.,10=V.TRAVAIL*/
    LIBELLE TEXTE 1
    E-R REFERE UN ENS-REL /* SI TYPE=1 OU 2 */
    P-VT REFERE UNE PROPRIETE  /* SI TYPE =3 OU 10 */
  FIN
FIN

/* ------------------------------------------------------------ */

/* ------------------------------------------------------------ */

ENTITE 1000 REGLE
DEBUT
  NOM MOT 8 AVEC CLE UNIQUE FIN
  RERREUR DE 1 A 2000
  ENTREE INVERSE TOUTE PRODUCTION DE UNE REGLE
  ENTITE 100 PRODUCTION
  DEBUT
    TERME REFERE UN VOCAB
    SUITE REFERE UNE PRODUCTION DE UNE REGLE
    PERREUR DE 1 A 2000
  FIN
FIN

/* ------------------------------------------------------------ */
```

```
ENTITE 1000 VOCAB
DEBUT
  NOM MOT 8 AVEC CLE UNIQUE FIN
  TYPE DE 1 A 4 /* 1 = REGLE VRAIE           */
                /* 2 = REGLE REGROUPEMENT    */
                /* 3 = MOT-CLE               */
                /* 4 = VALEUR A ENREGISTRER */
  DEFINITION REFERE UNE EXPLIC
  CONTROLE-SYNTAX MOT 8
  CONTROLE-SEMANT MOT 8
  EXECUT MOT 8
FIN

/* ------------------------------------------------------- */

ENTITE 20000 BATCH
DEBUT
  NUM DE 1 A 999999
    VAL MOT 30
FIN

/* ------------------------------------------------------- */

ENTITE 200 REGLE-EN-COURS /* PILE */
DEBUT
  REGLE REFERE UNE REGLE
  PRODUC REFERE UNE PRODUCTION DE UNE REGLE
  SY1 DE 0 A 1000000
  SY2 DE 0 A 1000000
  ACTIF DE 1 A 2 /* 1 = C'EST LA BONNE REGLE      */
                 /* 2 = C'EST SEULEMENT UN ESSAI */
FIN

/* ------------------------------------------------------- */

DICO-TOU INVERSE TOUT DICO
DICO-NOU INVERSE TOUT DICO
DICO-CRE INVERSE TOUT DICO
DICO-MAJ INVERSE TOUT DICO
DICO-MOD INVERSE TOUT DICO
DICO-PRO INVERSE TOUT DICO

/* ------------------------------------------------------- */

FORMAL CARTE
DEBUT
  FORMAL (80) CARAC
  DEBUT CAR MOT 1 FIN
  KARTE MOT (1 80)
/* ------------------------------------------------------- */
  NUMERO MOT (1 4)
  RANG MOT (5 7)
  LIBEL1 MOT (12 30)
  LIBEL2 MOT (42 30)
  C-OU-F MOT (72 1)
```

```
   OPERAT MOT (73 8)
 /* ------------------------------------------------------------ */
   CODCAR MOT (1 2)
   DATE MOT (3 8)
   CODNOM MOT (3 5)
   CODELM MOT (8 4)
   TYPOPE MOT (12 1)
   LIB1 MOT (13 30)
   LIB2 MOT (43 31)
 /* ------------------------------------------------------------ */
   AUTO-RES MOT ( 1  3)
   AUTO-TYP MOT ( 4  1)
   AUTO-NOM MOT ( 5  8)
   AUTO-ERR MOT (13  4)
   AUTO-SYN MOT (17  8)
   AUTO-SEM MOT (25  8)
   AUTO-EXC MOT (33  8)
   AUTO-BID MOT (41 38)
 /* ------------------------------------------------------------ */
 FIN
 /* ------------------------------------------------------------ */
 FORMAL TRI-DICO
 DEBUT
    PROJET DILATE 1
    CODE MOT 8
    INIS MOT 30
    NATU DILATE 2
 FIN
 /* ------------------------------------------------------------ */
 FORMAL TXT-BATCH
 DEBUT
   LOT DILATE 6
   LIGNE DILATE 6
   NB-LETTRES MOT 2
   VALEUR MOT 30
 FIN
 /* ------------------------------------------------------------ */
 FORMAL LIGNE
 DEBUT
   FORMAL (133) CARAC
   DEBUT
     CAR MOT 1
   FIN
   SAUT MOT (1 1)
   LIG1 MOT (2 80)
   LIG2 MOT (82 52)
 /* ------------------------------------------------------------ */
   CODEL MOT (1 4)
   ZONE1 MOT (5 30)
   ZONE2 MOT(35 30)
 FIN

 /* ------------------------------------------------------------ */

 FIN
```

ANNEXE B

QUELQUES PROGRAMMES DU PROJET MACSI-1

```
1     :DEFMAC SAISIE : : : :EXP
2     /* SAISIE DE :1: ET TRANSFERT DE :1: DANS :2:                    */
3     /* EN MODE BAVARD EDITION DU MESSAGE :3:                         */
4     /* CONTROLE DU CHANGEMENT DE MODE (BAVARD SILENCIEUX)            */
5     /* :1: = UNE CARACTERISTIQUE DE DICINIT                          */
6     /* :2: = UN Z                                                    */
7     /* :3: = NUMERO DU MESSAGE EXPLICATIF EN MODE BAVARD             */
8     /* MACRO UTILISEE : COMMENTAIRE                                  */
9     /* VARIABLES UTILISEES : Y20,Y21,Y22                             */
10    /* NOTA:SAISIE OUVRE UN FAIRE QUIL FAUDRA FERMER PAR UN          */
11    /* FIN A L'EXTERIEUR DE LA MACRO.                                */
12    EXEC TAPPEL1
13    M Y20 = 0 /* POUR REDEMANDES DANS LES MACROS DE CONTROLE        */
14    FAIRE
15    POUR X0
16       COMMENTAIRE :3:
17       M :1: DE UN DICINIT Y22 = U M :2: = U
18       M :1: DE UN DICINIT Y22 = EXT  /* SAISIE */
19       M :2: = :1: DE UN DICINIT Y22
20       SI :2: = 1/1 ALORS M Y21 = -Y21 REFAIRE FIN
21          /* CHANGEMENT DE MODE */
22    FIN
23    :FDEF
```

```
1     :DEFPRO CREEN  :CONTXT X0  :EXP
2     /* CREATION D'UN ENSEMBLE                                          */
3     /* CONCERNE TOPVAL,CODE,LIBELLE,DEFINITION,TAILLE,PROPR,          */
4     /*          CLES                                                  */
5     FAIRE
6                 D X1 = UN ENS-REL
7                 D X2 = UNE PROPRIETE
8                 D X8 = UN DICO
9                 D X9 = UNE EXPLIC
10        COMMENTAIRE 525
11        M Y5 = 2 /* ETAT : ERREUR */
12        ACCESCRE ENS-REL 21          /* X1 POINTE SUR UN ENS-REL*/
13                                     /* DE CODE Z1 SINON ERREUR */
14        SI Y25 = 1 ALORS DESASTRE SORTIE FIN
15        M Y5 = 1
16         M TYPE DE X1 = 1
17           COMMENTAIRE 501
18           /* SAISIE DE LA DEFINITION */
19           I 'DEFINITION :'
20           M Y25=0
21           EXEC GEX
22           SI EXISTE X9 ALORS M DEFINITION DE X1 = X9
23              M SUPPRESSION DE X9 = 1
24           FIN
25           M Y25 = 0
26           /* SAISIE DE LA TAILLE */
27           COMMENTAIRE 521
28           SAISIE TAILLE Z2 0
29                SI Z2 = 0 ALORS SORTIE FIN
30                M Y2 = 42   CONTROLE-PLAGE Y2 1 99999 379
31                 SI Y25 = 1 ALORS DESASTRE SORTIE FIN
32                 M TAILLE DE X1 = Y2
33           FIN
34              SI Y25 = 1 ALORS SORTIE FIN
35           M Y25 = 0
36           /* SAISIE  DES PROPRIETES */
37           COMMENTAIRE 526
38                I 'PROPRIETE : '
39                SAISIE CODE Z2 0
40                   M Y25 = 0
41                   NORMCOD Z2 Z3
42                   CONTREX Z2 549
43                   SI Y25 = 1 ALORS REFAIRE FIN
44                   SI Z2 = '$' ALORS  SORTIE FIN
45                   SI EXISTE UNE PROPRIETE AVEC CODE = Z2 ;
46                      ALORS ERREUR  576  REFAIRE
47                   FIN
48                   INIT PROPRIETE X2 22 Z2
49                    SI X2  = 0 ALORS
50                       M Y2 = NUMDE X2
51                       SI EXISTE UN PROPR Y2 DE X1
52                          ALORS ERREUR 532 REFAIRE
53                       FIN
54                       S UN PROPR DE X1 = X2
55                        M DEJUBI DE X2 = X1
56                   FIN
57                   REFAIRE
58                FIN
59          /* SAISIE DES CLES */                                                    . . .
```

```
60        FAIRE
61           M Y1 = 0 TOUT PROPR DE X1
62           SI Y1 = 0 ALORS SORTIE FIN
63          COMMENTAIRE 527
64                I 'CLE :'
65                SAISIE CODE Z2 0
66                   M Y25 = 0
67                   NORMCOD Z2 Z3
68                   CONTREX Z2 549
69                   SI Y25 = 1 ALORS REFAIRE FIN
70                   SI Z2 = '$$' ALORS SORTIE  FIN
71                   M X2 = UN PROPR        AVEC CODE = Z2 ; DE X1
72                   SI PAS    X2  ALORS  ERREUR 574   REFAIRE
73                   FIN
74                    M Y1 = NUMDE X2
75                    SI EXISTE UN KLE Y1 DE X1
76                       ALORS ERREUR 533 REFAIRE
77                    FIN
78                   G UNE KLE DE X1 = X2
79                   REFAIRE
80              FIN
81        FIN
82        FAIRE
83           TRAITE Z2
84           SI Z2 = 'R' ALORS EDIENS X1 REFAIRE FIN
85           SI Z1 = 'N' ALORS M Y5 = 2 /*ETAT ERREUR*/  FIN
86        FIN
87        SI Z1 = 'N' ALORS SORTIE FIN
88        BLOQUER
89           M Z1 = CODE DE X1
90           M X8 = UN DICO AVEC CODE = Z1 ;
91           M X9 = DEFINITION DE X1
92           GLED X8 X9
93        LIBERER
94        FIN-MAJ ENS-REL
95        FAIRE
96           SI Y5 = 1 ALORS SORTIE FIN
97           /* ERREUR FATALE : ON ANNULE CREEN */
98           BLOQUER
99              S DEFINITION DE X1
100             M Z1 = CODE DE X1
101             S UN DICO AVEC CODE = Z1 ;
102             S TOUT PROPR DE X1
103             S TOUT KLE DE X1
104             S X1
105          LIBERER
106          EFFACE
107       FIN
108       D X1  D X2  D X8  D X9
109    FIN
110    IFDEF
```

```
1     ;DEFPRO MAJCO :CONTXT X0 ;EXP .
2     /* MISE A JOUR DES COMMANDES APRES INITIALISATION                    */
3     D X1 = UNE COMMANDE
4     D X2 = UNE MODIF DE UNE COMMANDE
5     D X3 = UN DOCUMENT
6     D X8 = UN DICO
7     D X9 = UNE EXPLIC
8     D X7 = UN LISTE-DICO DE UNE EXPLIC
9     D X6 = UN LISTE-EXPLIC DE UN DICO
10    ACCESMAJ COMMANDE
11       SAISIE TYPE Z2 300
12       M Y1 = Z2
13          CONTROLE-PLAGE Y1 1 99 301
14       FIN
15       SI Y25 = 1
16          ALORS M CODE-EN-COURS DE UN DICINIT Y22 = 0
17                DESASTRE  M TOPVAL DE X1 = 2  SORTIE
18       FIN
19       M TYPE DE X1 = Y1
20       I 'DOCUMENT :'
21       SAISIE CODE Z2 302
22          NORMCOD Z2 Z3
23          SI Z2 = '**' ALORS SORTIE FIN
24          INIT DOCUMENT X3 12 Z2
25          SI X3  = 0
26             ALORS 0 UN DOC-ASSOCIE DE X1 = X3
27          FIN
28          REFAIRE
29       FIN
30       I 'DEFINITION :'
31       COMMENTAIRE 303
32       M Y1 = 0
33       FAIRE
34          EXEC GEX
35          SI X9  = 0  ALORS M Y1 = 0  SORTIE  FIN
36          ERREUR 308
37          SI Y1 = 1  ALORS  SORTIE  SINON M Y1 = 1 REFAIRE FIN
38       FIN
39       SI Y1 = 1              /* SI PAS D EXPLIC ON ANNULE TOUT */
40          ALORS M TYPE DE X1 = 0
41                SE TOUT DOC-ASSOCIE DE X1
42                EFFACE  DESASTRE
43                M CODE-EN-COURS DE UN DICINIT Y22 = 0
44                M TOPVAL DE X1 = 2     SORTIE
45       FIN
46       M TOPVAL DE X9 = 1
47       M DEFINITION DE X1 = X9
48       M DATE-1 DE X1 = DATE DE UN DICINIT Y22
49       M ETAT DE X1 = 1
```

• • •

```
50        FAIRE
51           TRAITE Z7
52           SI Z7 = INT
53              ALORS EDICO X1 X2 X3 Y10  REFAIRE
54           FIN
55        FIN
56        SI Z7 = INT
57           ALORS S. X9
58                 SE TOUT. OBJ.ASSOCIE DE X1
59                 EFFACE
60                 M TYPE DE X1 = J   M ETAT DE X1 = J
61                 I (4) '** ANNULATION EFFECTIVE **'
62                 M CODE-ENCOURS DE UN DICINIT Y22 = J
63                 M TOPVAL DE X1 = 2   SORTIE
64        FIN
65        /* MAJ DE L EXPLIC */
66        M MERA DE X9 = MERA DE X1
67        M Z2 = VOTRE-CODE DE UN DICINIT Y22
68        M AUTEUR DE X9 = UNE PERSONNE AVEC CODE = Z2 ;
69        M SUPPRESSION DE X9 = 1
70        A UN LISTE-DICO X7 DE X9
71        M VAL DE X7 = X8
72        /* MAJ DU DICO */
73        A UN LISTE-EXPLIC X6 DE X8
74        M NUM-EDITION DE X6 = 1
75        M VAL DE X6 = X9
76     FIN-MAJ COMMANDE
77     D X1 D X2 D X3 D X6 D X7 D X8 D X9
78     .FDEF
```

```
1     :DEFPRO MODPP          :CONTXT X0          :EXP
2     /* MODIFICATION DES PROPRIETES APRES MAJ                          */
3     /* CONCERNE  TOPVAL,LIBELLE,DEFINITION,CONTROLE,TYPE,             */
4     /*           COMPOSANTS                                           */
5     /* AUTORISE    LIB , DEF , CTL , TYPE , COMP- , COMP+             */
6     FAIRE
7               D X1 = UNE PROPRIETE
8               D X2 = UNE PROPRIETE
9               D X8 = UN DICO
10              D X9 = UNE EXPLIC
11              D X4 = UN ENS-REL
12              D X3 = UN ENS-REL
13              D X5 = UNE EXPLIC
14              D X7 = UNE EXPLIC
15        COMMENTAIRE 510
16        ACCESMOD PROPRIETE      /* X1 POINTE SUR UNE PROPRIETE      */
17                                /* DE CODE Z1 SINON DESASTRE        */
18                                /* X8 POINTE SUR LE DICO            */
19           M X3 = DEQUOI DE X1
20           M X5 = DEFINITION DE X1
21           M X7 = PPCTRL DE X1
22           SAISIE MODIF-DE Z2 0     /* Z2=MOT-CLE DE LA MODIF      */
23           M Y25 = 0
24                SI Z2 = 0 ALORS REFAIRE FIN
25                NORMCOD Z2 Z3
26                SI Z2 = 'RECAP' ALORS EDIPP X1  REFAIRE
27                FIN
28                SI Z2 = 'SS' ALORS SORTIE FIN
29                SI Z2 = 'LIB' ALORS JLIB 594 REFAIRE FIN
30               SI Z2 = 'DEF' ALORS JDEF DEFINITION X1 REFAIRE FIN
31                SI Z2 = 'CTL' ALORS JDEF PPCTRL X1 REFAIRE FIN
32                SI Z2 = 'TYPE' ALORS MOTYPP REFAIRE FIN
33                SI TYPE DE X1 = 1 ALORS
34                   SI Z2 = 'COMP+' ALORS MOCPP REFAIRE FIN
35                   SI Z2 = 'COMP-' ALORS MOCPM REFAIRE FIN
36                FIN
37                ERREUR 595
38                REFAIRE
39           FIN                /* DE SAISIE */
40        FIN-MAJ PROPRIETE
41        D X1 D X2 D X8 D X9 D X4   D X3 D X7 D X5
42    FIN
43    :FDEF
```

ANNEXE C

UNE UTILISATION DE MACSI-1 EN MODE CONVERSATIONNEL

```
PROJET:M
FRAPPEZ L'OPERATION QUE VOUS VOULEZ FAIRE:CRE POUR CREATION,
MAJ POUR MISE A JOUR,MOD POURMODIFICATION SUIVI DES DEUX
PREMIERES LETTRES DE L'ENTITE...
OPERATION:
```

mode BAVARD

```
CRESE
VOUS ETES EN CREATION DE SERVICE ADMINISTRATIF , AU COURS
DE SA CREATION VOUS ETES LA SEULE PERSONNE QUI PUISSE Y ACCE
DER AINSI QU'A SES EMPLOYES....
CODE:
CODE:DIR01
DEB-LIBEL:DIRECTION DE  L'ETABLISSEMENT
FIN-LIBEL:SCOLAIRE
LA DEFINITION EST UN TEXTE LIMITE A CENT LIGNES DE SOIXANTE
CARACTERES CHACUNE...
DEFINITION :
-  :                              ....:
D-L:LA DIRECTION DE L'ETABLISSEMEN
F-L:T EST LE SERVICE LE PLUS HAUT
-  :                              ....:
D-L:DANS LA HIERARCHIE DE L'ETABLI
F-L:SSEMENT.
-  :                              ....:
D-L:/
D-L:IL EST CHARGE DE LA DIRECTION
F-L:ET DE L'ADMINISTRATION DE L'ET
D-L:ABLISSEMENT
F-L:
* TEXTE  504 *
DEB-ADRES:5 RUE TRISTAN BERNARD
FIN-ADRES:ST MARTIN D'HERES
PAYS:FRANCE
TELEPHONE:
TELEX:/
S'IL N'Y A PAS DE TELEX : TAPER RAS , S'IL EXISTE ET QUE
VOUS NE LE CONNAISSEZ PAS FAITES UN RETOUR CHARIOT
TELEX:
POUR CHAQUE EMPLOYE  SERONT PRECISES: LE NOM,LE PRENOM.POUR
ACHEVER LA LISTE DES EMPLOYES,TAPER  DEUX $ ($$)  A L'APPA
RITION DE 'EMPLOYE :'...
EMPLOYE:/
EMPLOYE:P007
NOM:DUPONT
PRENOM:JEAN
D'ACCORD ?
REPONSE:O
EMPLOYE:P015
NOM:SCHAEL
PRENOM:COCO
D'ACCORD ?
REPONSE:/
```

PASSAGE en SILENCIEUX

PASSAGE en BAVARD

PASSAGE en SILENCIEUX

PASSAGE en BAVARD

REPONSE:O
POUR CHAQUE EMPLOYE SERONT PRECISES: LE NOM,LE PRENOM.POUR
ACHEVER LA LISTE DES EMPLOYES,TAPER DEUX $ (S$) A L'APPA
RITION DE 'EMPLOYE :'...
EMPLOYE:SS
LA DEMANDE DE TRAITEMENT VOUSPERMET, SOIT D'INTERROMPRE LE
TRAITEMENT EN COURS (REPONDRE N), SOIT DE VALIDER TOUTES
INFORMATIONS FOURNIES (REPONSEO) ET AUTORISER LA PRISE
EN COMPTE, SOIT DE DEMANDER UNRESUME (REPONSE R) AVANT DE
DECIDER....
ENREGISTREMENT:/ ← PASSAGE en SILENCIEUX
ENREGISTREMENT:R
↑ DEMANDE de RECAPITULATION

***

SERVADM : DIR01

CODE : DIR01

LIBELLE :
DIRECTION DE L'ETABLISSEMENTSCOLAIRE

DEFINITION :
LA DIRECTION DE L'ETABLISSEMENT EST LE SERVICE LE PLUS HAUT
DANS LA HIERARCHIE DE L'ETABLISSEMENT.
IL EST CHARGE DE LA DIRECTION ET DE L'ADMINISTRATION DE L'ET
ABLISSEMENT...

EMPLOYE :

P007 DUPONT
JEAN
P015 SCHAEL
COCO

PAYS : FRANCE
ADRESSE :
5 RUE TRISTAN BERNARD_ST MARTIN D'HERES
TELEPHONE : ...
TELEX : ...

***

ENREGISTREMENT:O ← VALIDATION
ACCES:PRIVE

VOUS VENEZ D'INITIALISER:
DUPONT JEAN
SCHAEL COCO

** FIN CREATION SERVADM DIR01 **

```
OPERATION:CREPE
CODE:P009
NOM:TARTAMPION
PRENOM:LOUIS-PHILIPPE
ADMINISTRATION:DIR01
DEB-ADRES:10 RUE BICONDE
FIN-ADRES: 38 - GRENOBLE -
PAYS:FRANCE
TELEPHONE:
ENREGISTREMENT:R
```

← RECAPITULATION

```
***

PERSONNE : P009

CODE          : P009
NOM           : TARTAMPION
PRENOM        : LOUIS-PHILIPPE

ADMINISTRATION : DIR01
DIRECTION DE  L'ETABLISSEMENTSCOLAIRE

ADRESSE       :
10 RUE BICONDE 38 - GRENOBLE -

PAYS          : FRANCE
TELEPHONE     : ...

              ***
ENREGISTREMENT:O
ACCES:PRIVE

** FIN CREATION PERSONNE P009 **
```

```
OPERATION:CREEN
CODE:E1
DEB-LIBEL:PROFESSEURS
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
DEFINITION :
D-L:/
-   :                                ....:
D-L:/
D-L:ENSEMBLE DE PERSONNES QUI ASSU
F-L:ENT UN POSTE D'ENSEIGNANT DANS
D-L:L'ETABLISSEMENT SCOLAIRE. ILS
F-L:DOIVENT ETRE TITULAIRES DU POS
D-L:TE OU VACATAIRES ET ASSURER UN
F-L:MINIMUM DE 60 HEURES D'ENSEIGN
D-L:EMENT PAR ANNEE SCOLAIRE
F-L:
* TEXTE  508 *
TAILLE:50
PROPRIETE :
CODE:P11
DEB-LIBEL:NOM
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P12
DEB-LIBEL:PRENOM
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P13
DEB-LIBEL:NUMERO SECURITE SOCIALE
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P17
DEB-LIBEL:CATEGORIE
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:$$
CLE :
CODE:P13
CODE:$$
ENREGISTREMENT:R
```

RECAPITULATION

```
**********
*ENSEMBLE*
**********

CODE      : E1
LIBELLE   : PROFESSEURS          ...
DEFINITION:
...                              ...
L'ETABLISSEMENT SCOLAIRE. ILS DOIVENT ETRE TITULAIRES DU POS
TE OU VACATAIRES ET ASSURER UNMINIMUM DE 60 HEURES D'ENSEIGN
EMENT PAR ANNEE SCOLAIRE         ...
TAILLE    : 50

PROPRIETES :
CODE:P11
CODE:P12
CODE:P13
CODE:P17

CLES      :
CODE:P13
ENREGISTREMENT:0
ACCES:PUBLIC

VOUS VENEZ D'INITIALISER:
PROPRIETE:P11
NOM...
PROPRIETE:P12
PRENOM...
PROPRIETE:P13
NUMERO SECURITE SOCIALE...
PROPRIETE:P17
CATEGORIE...


** FIN CREATION ENS-REL E1 **
```

```
OPERATION:MAJPP
CODE:P13
NUMERO DE SECURITESOCIALE
DEFINITION :
D-L:LE NUMERO DE SECURITE SOCIALE
F-L:PERMET L'IDENTIFICATION SANS AM
D-L:BIGUITE DES PROFESSEURS AINSI
F-L:QU'UN CRITERE DE TRI
D-L:
* TEXTE  75 *
TYPE:1
CONTROLE :
D-L:IL FAUDRA CONTROLER SON EXISTE
F-L:NCE , IL DOIT ETRE FORME PAR
D-L:13 CHIFFRES COMMENCANT TOUJOURS
F-L: PAR 1 OU 2
D-L:
* TEXTE  76 *
PROPRIETES COMPOSANTES :
CODE:P131
DEB-LIBEL:SEXE
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P132
DEB-LIBEL:ANNEE
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P133
DEB-LIBEL:MOIS
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P134
DEB-LIBEL:DEPT
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P135
DEB-LIBEL:COMMUNE
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:P136
DEB-LIBEL:ORDRE
FIN-LIBEL:
D"ACCORD ?
REPONSE:O
CODE:$$
ENREGISTREMENT:R
```

```
**********
*PROPRIETE*
**********

CODE       :P13
LIBELLE    :NUMERO DE SECURITE SOCIALE
DEFINITION:
LE NUMERO DE SECURITE SOCIALE PERMET L'IDENTIFICATION SANS A
BIGUITE DES PROFESSEURS AINSI QU'UN CRITERE DE TRI
ENSEMBLE   :E1
TYPE       : 1

COMPOSANTE DE :
COMPOSEE DE    :
CODE:P131
CODE:P132
CODE:P133
CODE:P134
CODE:P135
CODE:P136
CONTROLE   :
IL FAUDRA CONTROLER SON EXISTENCE , IL DOIT ETRE FORME PAR
13 CHIFFRES COMMENCANT TOUJOUR PAR 1 OU 2
ENREGISTREMENT:O
ACCES:PRIVE

VOUS VENEZ D'INITIALISER:
PROPRIETE:P131
SEXE...
PROPRIETE:P132
ANNEE...
PROPRIETE:P133
MOIS...
PROPRIETE:P134
DEPT...
PROPRIETE:P135
COMMUNE...
PROPRIETE:P136
ORDRE...


** FIN MISE A JOUR PROPRIETE P13 **
```

```
OPERATION:MAJCO
CODE:C18
CONTROLES DIRECTS ET INDIRECTS...
TYPE:1
DOCUMENT :
CODE:D07
DEB-LIBEL:ETAT DES ANOMALIES
FIN-LIBEL:
D'ACCORD ?
REPONSE:O
CODE:D43
DEB-LIBEL:BORDEREAU DE PERFORATION
FIN-LIBEL:
D'ACCORD ?
REPONSE:O
CODE:$$
DEFINITION :
D-L:CETTE COMMANDE EST CHARGE DE
F-L: FAIRE DES CONTROLES DIRECTES
D-L: SUR LES NOTES (NUMERIQUES ET
F-L: ENTRE 0 ET 20) SUR LE CODE
D-L: DE MATIERS (NUMERIQUE), SUR
F-L: LE TRIMESTRE EN COURS (NUMER
D-L:IQUE ET ENTRE 1 ET 4), ET DE
F-L: FAIRE DES CONTROLES INDIRECT
D-L:S EN VERIFIANT QUE L'ELEVE EX
F-L:ISTE BIEN DANS LA\E FICHIER
D-L: DES ETUDIANTS. UN DOCUMENT
F-L: CONTENANT LES ANOMALIES CONS
D-L:TATES SERA EDITE PAR LA COMMA
F-L:NDE
D-L:
* TEXTE  524 *
ENREGISTREMENT:R

COMMANDE : C18   MACSI
CONTROLES DIRECTS ET INDIRECTS...
TYPE :  1
DEFINITION: EXPLIC NO.:  524
DOCUMENTS ASSOCIES :
CODE : D07
ETAT DES ANOMALIES...
CODE : D43
BORDEREAU DE PERFORATION...
ETAT :  1
MODULE(S) UTILISATEUR(S) :
CODE : M10
EXAMENS TRIMESTRIELS...
GROUPON(S) EXECUTANT(S) :
CODE : G5
EQUIPE DE EXPLOITATION...

ENREGISTREMENT:O
ACCES:PRIVE

VOUS VENEZ D'INITIALISER:
DOCUMENT:D07
ETAT DES ANOMALIES...
DOCUMENT:D43
BORDEREAU DE PERFORATION...
```

[8] J. COURTIN, J. VOIRON

Introduction à l'algorithmique et aux structures de données.
Cours polycopié - I.U.T.-B - Département Informatique.
Université des Sciences Sociales de Grenoble. 1974.

[9] CXP

Etude comparative des méthodes d'analyse.
Centre d'eXpérimentation des Packages - Etude 008.

[10] O.J. DAHL, E.W. DIJKSTRA, C.A.R. HOARE

Structured Programming.
Academic Press. 1972.

[11] C. DELOBEL

Contributions théoriques à la conception et l'évaluation d'un système d'informations appliqué à la gestion.
Université Scientique et Médicale de Grenoble.
Thèse d'Etat. Octobre 1973.

[12] E.W. DIJKSTRA

A short Introduction to the Art of Programming.
EWD316. University of Eindhoven. 1971.

[13] EDP Analyser

The current status of data management.
Février 1974.

[14] J.P. GIRAUDIN

OASIS. Un outil d'analyse structurée.
Rapport de D.E.A.. Septembre 1974.
Université Scientifique et Médicale de Grenoble.
Laboratoire d'Informatique.

[15] GRADIA

Le projet MEDOC.
Note interne Mai 1975.
Groupe d'Appui pour le Développement de l'Informatique dans l'Administration - 27, Quai Anatole France - PARIS (7e).

[16] M. GRIFFITHS

Analyse syntaxique pour la production de compilateurs.
Cours polycopié. Université Scientique et Médicale
de Grenoble. Laboratoire d'Informatique. Octobre 1973.

[17] J.J. HORNING et B. RANDELL

Process Structuring.
ACM Computing Surveys. Vol. 5 n° 1. Mars 1973.

[18] INFORMATIQUE et GESTION N° 45

Dossier : les méthodes d'analyse.
(Sont succinctement présentées les méthodes METALOG,
MINOS, ARMIN, PARM, PROTEE, ARIANE).

[19] P. MAURICE, P.C. SCHOLL

Un interpréteur du langage APL.
Thèse de Docteur-Ingénieur. Université de Toulouse.
Octobre 1970.

[20] H.D. MILLS

The Impact of Structured Programming on Software Engineering
and Production.
Chapitre français de l'ACM. Février 1973.

[21] F. NICK, C.P. OHLEN, V. SCHAEDEL, R. SCHAEFER

Systeme der computergestützen Systemgestaltung.
BIFOA Arbeits bericht 1972.
WISON VERLAG KOLN.

[22] F. PECCOUD

MACSI : Méthode d'Aide à la Conception des Systèmes d'Informations.
Université Scientifique et Médicale de Grenoble.
Thèse d'Etat. Juin 1975.

BIBLIOGRAPHIE

[1] J.R. ABRIAL

Projet Socrate.

Cours Avancé sur l'Architecture des Systèmes.

Alpe d'Huez - Décembre 1972.

[2] J.R. ABRIAL

Data Semantics.

Université Scientifique et Médicale de Grenoble.

Laboratoire d'Informatique. Novembre 1973.

[3] J.D. ARON

Information System in Perspective.

Computing Surveys Vol. 1, Décembre 1969.

[4] A. de CHELMINSKI

Método de Ayuda a la Concepcion de Sistemas de Información.

I Simposium Venezolano sobre Ingenieria de

Sistemas. CARACAS. Avril 1975.

[5] Y. CHIARAMELLA

Cobol - présentation d'un sous-ensemble du langage
- éléments de programmation structurée.

Cours polycopié. I.U.T.-B - Département Informatique.

Université des Sciences Sociales de Grenoble. 1974.

[6] M.E. CONWAY

Design of a Separable Transition-Diagram Compiler.

ACM Vol. 6, Number 7. Juillet 1963.

[7] J.D. COUGER

Evaluation of Business System Analysis Techniques.

ACM Computing Surveys, Vol. 5 n° 3, Septembre 1973.

[23] F. PECCOUD

Syntactic and Semantic structures of files.

File 68 - Sponsored by IFIP administrative data processing group.

Copenhague. Novembre 1968.

[24] F. PECCOUD

Réflexions sur la construction d'un modèle d'aide à la conception des systèmes d'information.

Ecole d'Eté de l'AFCET - Alès, Juillet 1971.

[25] F. PECCOUD et al.

MACSI : un modèle d'aide à la conception des systèmes informatiques.

Colloque "Les outils de l'analyse en informatique de gestion". Universités de Grenoble - Octobre 1971.

[26] F. PECCOUD

Ein Modell für die interacktive computergestützte Entwicklung grosser Informationssysteme.

BIFOA. Köln - Août 1973.

[27] SOCRATE

Manuel d'utilisation - Manuel d'opération.

ERIA - ECA Automation - Juillet 1974.

[28] SOCRATE SIRIS 7/SIRIS 8

Manuel d'utilisation - Manuel d'opération.

CII - Juillet 1973.

[29] SOCRATE SIRIS 2/SIRIS 3

Manuel d'utilisation - Manuel d'opération.

CII - Juillet 1973.

[30] A. STIERS

Manuel utilisateur SOCRATE - Version 4.

[31] D. TEICHROEW

A Survey of languages for stating requirements for computer-based information systems.

Fall Joint Computer Conference 1972.

[32] D. TEICHROEW, E.A. HERSHEY, M.J. BASTARACHE

An Introduction to PSL/PSA.

ISDOS WORKING paper N° 86 - Mars 1974.

The University of Michigan - Ann Arbor Michigan 48104.

[33] B. VAUQUOIS

Calculabilité des Langages.

Cours polycopié. C3 Maîtrise d'Informatique.

Université Scientifique et Médicale de Grenoble.

Laboratoire d'Informatique. 1970.