Méthodes et outils pour la production de didacticiels : l'environnement informatique du projet MOSAIQUE

Jean-Michel Adam

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE
Informatique

par

Jean-Michel ADAM

METHODES ET OUTILS POUR LA PRODUCTION DE DIDACTICIELS : L'ENVIRONNEMENT INFORMATIQUE DU PROJET MOSAIQUE

Thèse soutenue le 8 décembre 1983 devant la Commission d'Examen :

Monsieur	G. VEILLON	:	Président
Madame	M. QUERE		Examinateurs
Messieurs	S. KRAKOWIAK		
	M.MILCHBERG		
	P.C. SCHOLL		

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENÉVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C,N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

Je tiens à remercier Monsieur Gérard VEILLON, Professeur à l'I.N.P.G., Directeur de l'E.N.S.I.M.A.G., pour l'honneur qu'il me fait de présider le jury de cette thèse.

Je remercie Madame Maryse QUERE, Professeur à l'I.U.T. de Nancy 2, d'avoir accepté de participer à ce jury.

Je remercie également Monsieur Sacha KRAKOWIAK, Professeur à l'U.S.M.G., d'avoir bien voulu s'intéresser à cette thèse.

Monsieur Pierre-Claude SCHOLL, Professeur à l'U.S.M.G. a constamment suivi mes travaux de recherche, depuis leur début. Je le remercie particulièrement de m'avoir accueilli dans son équipe et d'avoir accepté la responsabilité de ces recherches. Je tiens à souligner le temps important qu'il a consacré à mon travail : sa rigueur et son esprit critique, au cours de nombreuses discussions de fond, m'ont beaucoup apporté.

Monsieur Mauricio MILCHBERG, Professeur à l'U.S.M.G., a suivi de près, durant ces deux dernières années, le déroulement du projet MOSAIQUE. Je le remercie sincèrement pour ses conseils précieux et ses remarques pertinentes durant la rédaction de ce document. Son réalisme et la qualité de son jugement ont été déterminants dans la concrétisation de cette thèse.

Je remercie Michel SIMONET, Chercheur au C.N.R.S., et Michel DELAUNAY, Ingénieur au C.N.R.S., d'avoir accepté de lire certaines parties techniques de la thèse. En tant que spécialistes, leurs remarques m'ont été d'une aide importante.

Dominique ABRY et Maryse SCHOLL, enseignantes de Français - langue étrangère au C.U.E.F., ont constitué l'équipe pédagogique du projet. Leur imagination débordante a été la source de nombreuses propositions, qui ont contribué à la qualité des didacticiels et mis à l'épreuve les outils que nous avions créés. Je les remercie vivement de leur coopération.

Je tiens également à remercier particulièrement Sylvie PAINVIN, psycho-pédagogue, qui a apporté par son expérience en E.A.O. et ses critiques contructives, la qualité aux didacticiels produits. Elle a constitué le lien entre l'équipe pédagogique et l'équipe informatique. Ses qualités d'organisatrice ont fortement contribué à la bonne marche du projet.

Catherine DOYAT, Marie-Pierre FLEURY, Yves COSTE et Hubert DELORI ont constitué avec moi l'équipe informatique. La diversité de leurs expériences a apporté au projet une richesse de compétences qui fut déterminante pour mener à bien notre tâche. Je les remercie pour la qualité du travail qu'ils ont effectué, et leur enthousiame de tous les instants.

Je remercie également Georges FAFIOTTE, enseignant à l'U.S.S.G., pour son concours en tant qu'expert en E.A.O. et spécialiste des matériels informatiques, et pour les longues et fructueuses discussions que nous avons eues ensemble.

Les autres membres de l'équipe se sont intéressés dès le début au projet. Je les en remercie sincèrement, ainsi que pour leur bonne humeur qui contribua à la bonne ambiance de l'équipe.

Enfin, je tiens à remercier mes amis Geneviève BICAIS, Danielle ZIEBELIN, Philippe GENOUD, Véronique et Alain LUCCI, Sophie LEMOINE, Luc MAS, Bruno DEFUDE, Yves COSTE, Christine DUBONNET, Jacques BAYLACQ, Marie-Pierre FLEURY, Richard DURAND, Richard DESBIOLLES, Catherine THOLLON et Daniel CHOVIN (cités sans ordre de préférence), pour leur aide très précieuse dans la frappe et la mise en forme de cette thèse.

Je souhaite que le lecteur apprécie la qualité de la réalisation pratique du document, due au service tirage du laboratoire.

Jean-Michel ADAM

Sommaire

INTRODUCTION

L'utilisation de l'ordinateur à des fins pédagogiques a vu le jour aux U.S.A. dans les années 1960. Après des débuts marginaux en France dès 1967 (Paris, Grenoble), on assiste depuis quelques années à un développement de l'Enseignement Assisté par Ordinateur (E.A.O.), et des expériences sont effectuées dans le cadre de l'Education Nationale et dans quelques grandes entreprises [Sim 80] [Bos 82].

La spécification de logiciels d'enseignement (didacticiels) pose aux pédagogues des problèmes dûs à leur manque de connaissances en informatique. Aussi, voit-on se développer des outils qui favorisent cette spécification : langages auteurs, systèmes auteurs [Kea 82] [ADI 83].

Le projet MOSAIQUE s'inscrit dans le cadre d'une recherche de méthodes et d'outils qui puissent dégager les auteurs des contraintes de l'informatique, et permettre une production de programmes fiables, facilement modifiables et transportables sur des environnement de micro-ordinateurs.

Le projet MOSAIQUE

Les objectifs généraux du projet MOSAIQUE [ScF 80] [Sch 80] étaient les suivants :

- concevoir et développer des méthodes et des outils de création de didacticiels, avec comme terrain d'application l'E.A.O. du Français langue étrangère (F.L.E.) ;

- produire, en appliquant ces méthodes et en utilisant ces outils, un didacticiel de soutien dans l'enseignement du français à des étudiants arabophones.

Les moyens nécessaires à la mise en oeuvre du projet ont pu être obtenus grâce à une participation financière importante de l'Agence de l'Informatique (A.D.I.), qui permit le recrutement de personnel, l'achat de matériel informatique et audio-visuel, et le fonctionnement de l'équipe ainsi formée. Ces moyens ont été complétés par un ensemble de ressources locales (humaines, structurelles et matérielles).

Les ressources humaines locales furent constituées de quatre sortes de compétences : expérience en méthodologie de la programmation [Sch 79] et génie logiciel, expérience en enseignement assisté par ordinateur ([FPS 76], [Faf 76], [FFG 78], [Faf 78]), expérience en enseignement du français - langue étrangère [ScS 72], expérience en électronique et matériels audio-visuels [Faf 81]. Vinrent s'y ajouter des compétences diverses provenant de l'environnement grenoblois : un important laboratoire de recherche en informatique, un centre d'enseignement du français - langue étrangère, un institut de didactique des langues, un centre de formation à l'informatique de professeurs de l'enseignement secondaire.

A des titres divers et sur des durées différentes, une dizaine de personnes ont participé à ce projet :

- deux professeurs de F.L.E. détachés de leurs postes,
- une psychopédagogue,
- un linguiste arabophone,
- cinq informaticiens dont un chercheur et quatre programmeurs experts.

De plus, trois enseignants-chercheurs sont intervenus sur le projet à temps partiel.

Déroulement du projet :

Le matériel utilisé est composé

- d'une configuration de "diffusion" destinée à supporter les didacticiels produits, elle comprend :

  . un micro-ordinateur à microprocesseur 8 bits de 64 K octets de mémoire vive, muni de lecteurs de disquettes 8 pouces,
  . deux périphériques audio visuels : magnétophone à cassettes et projecteur de diapositives,
  . une imprimante.

- d'une configuration de "production" plus puissante, destinée à la production des didacticiels, elle comprend :

  . un micro-ordinateur de 256 K octets de mémoire, muni d'un disque dur de 28 M octets et d'un microprocesseur de 16 bits. Il est géré par un système d'exploitation multi-tâches, multi-utilisateurs.
  . cinq terminaux et une imprimante pouvant y être connectés.

Une formation à l'E.A.O. et une information réciproque sur les compétences de chacun ont été menées pendant les six premières semaines du projet au sein de l'équipe [Pai 81]. A la fin de cette formation initiale, une maquette du projet a été réalisée.

Trois grandes phases peuvent être distinguées dans le déroulement du projet :

. La première phase se situe de Janvier 1981 à Avril 1982. Elle a consisté à réaliser, sur micro-ordinateur, un premier didacticiel de F.L.E. qui porte sur l'enseignement du passé-composé [SAP 81]. Il est composé de 40 exercices et fait intervenir des médias audio-visuels et conventionnels que pilote le micro-ordinateur : un projecteur de diapositives, un magnétophone à cassettes et une imprimante.

Cette première réalisation a permis de mettre en évidence des problèmes :

- au niveau de la spécification du didacticiel par les pédagogues,
- au niveau de la réalisation informatique.

Au cours de cette période, un ensemble de méthodes pour la production de didacticiels ont été définies [Adm 81] et expérimentées. Elles ont permis de définir les fonctionnalités d'un environnement de production de didacticiels.

. La **deuxième phase** se situe de mai à décembre 1982. Elle a donné lieu à la réalisation et la mise au point par l'équipe d'informaticiens, d'un ensemble d'outils qui favorisent la production de didacticiels [ACD 81]. Ces outils ont été expérimentés pour la réalisation d'un second didacticiel qui a pu être testé et mis au point par les auteurs sur le site de production, avant son transfert sur le site de diffusion. Dans le même temps, un système de production manuelle ou automatique de cassettes de messages sonores a été spécifié, et sa maquette réalisée [FaI 82]. Parallèlement, les pédagogues ont spécifié des exercices sur le futur et le présent, en appliquant les méthodes définies au cours de la première phase du projet.

En novembre 1982, le projet fut réorienté vers la réalisation d'un didacticiel de "large diffusion".

. La **troisième phase** se situe de décembre 1982 à avril 1983. Elle a consisté à réaliser un didacticiel de "grande diffusion" portant sur les trois temps fondamentaux du discours : le passé-composé, le présent et le futur. La configuration matérielle supportant ce didacticiel est plus légère : un micro-ordinateur et un magnétophone à cassettes bas de gamme piloté par l'apprenant. Cette réalisation a nécessité un remaniement important des exercices, tant sur le plan pédagogique qu'informatique, la synchronisation des différents messages visuels et sonores n'étant plus réalisable par ordinateur. Elle a permis de vérifier l'adéquation des méthodes définies et des outils réalisés, aux problèmes de la production.

Le travail

Cette thèse a pour objet :

a) **De proposer des méthodes pour la production de didacticiels :**

Sur la base d'un modèle des fonctions essentielles requises pour l'écriture d'un didacticiel (chapitre 1), nous étudions les problèmes posés par la spécification de ces fonctions. Pour chaque fonction, nous étudions les informations que l'auteur doit définir, pour éviter les ambiguités et fournir une spécification complète à l'informaticien (chapitre 2).

Nous étudions les problèmes de la réalisation du programme informatique correspondant aux spécifications de l'auteur (chapitre 3). Nous proposons des méthodes de description de didacticiel dans une notation algorithmique, et des règles de passage de cette description à un programme informatique.

Nous nous intéressons aux problèmes d'implantation des programmes produits, sur des micro-ordinateurs, ainsi qu'à leur portabilité sur des sites de même type. Enfin, nous étudions les problèmes posés par la modification de ces programmes.

Les méthodes proposées sont issues de l'expérience concrète et réussie de production de didacticiels par l'équipe de travail du projet MOSAIQUE.

b) **De rendre compte d'une expérience de génie logiciel en vraie grandeur :**

Nous décrivons l'ensemble des outils que nous avons dû spécifier et réaliser pour favoriser la production de didacticiels (chapitre 4). Ces outils permettent, à partir d'une description de didacticiel dans la notation algorithmique, de produire les programmes correspondants, exécutables sur un site donné. Ils facilitent la modification et la gestion des informations et des programmes manipulés. Ils favorisent leur implantation sur des micro-ordinateurs et leur transport sur de nouveaux matériels.

Mon rôle au sein du projet MOSAIQUE a consisté à effectuer le **travail de recherche** sur le plan informatique, c'est à dire, dans un premier temps, à étudier les caractéristiques des didacticiels ; cette étude s'est faite parallèlement au travail de la commission E.A.O. de l'Agence de l'Informatique, à l'issue duquel fut proposée une structuration des didacticiels [ADI 81].

Cette première étude m'a servi de base à un travail de **définition de méthodes** de programmation de didacticiels. Une première expérimentation de ces méthodes (première phase du projet) a donné lieu à la réalisation manuelle d'un premier didacticiel de taille importante. Ce travail me permit :

. de dégager les problèmes essentiels dans la spécification de didacticiels par les auteurs,
. de **spécifier un ensemble d'outils** qui favorisent la production des didacticiels.

Les outils proposés ont été **réalisés et validés** à travers la production d'un second didacticiel (seconde phase du projet).

A ce travail de recherche s'ajoute un travail de coordination de l'équipe informatique, ainsi que ma participation aux réalisations techniques proprement dites :

- la réalisation d'une maquette du projet,
- l'écriture de fonctions de bases nécessaires à la programmation des didacticiels, notamment au niveau de la gestion des fichiers,
- l'écriture d'un "moniteur" supportant le didacticiel sur le site de diffusion,
- la réalisation partielle du noyau du système de production,
- la réalisation d'une interface qui permette le recouvrement de segments de programmes en mémoire sur le site de diffusion.

CHAPITRE I

STRUCTURATION D'UN DIDACTICIEL

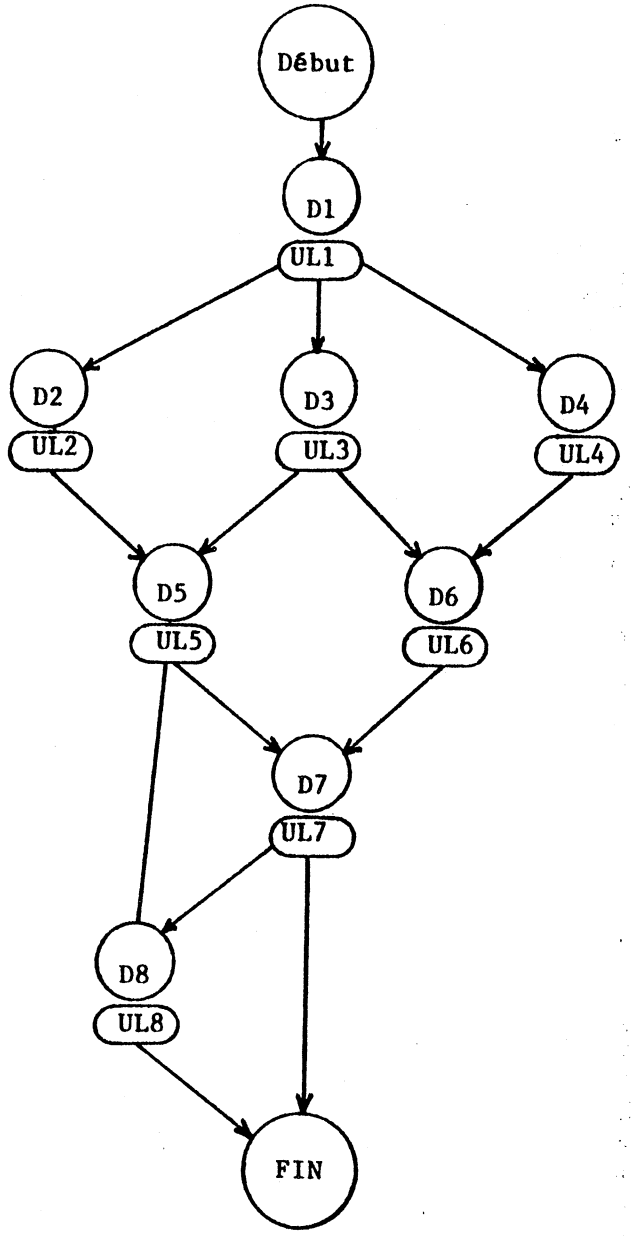
Un didacticiel est un logiciel d'enseignement traitant un sujet pédagogique global. Le projet MOSAIQUE a donné lieu à la réalisation de deux didacticiels en Français - Langue Etrangère. Les normes utilisées pour leur structuration ont été définies à partir de celles proposées par la commission E.A.O. de l'Agence de l'Informatique ([ADI 81] repris dans [BeF 82]). Dans cette première partie, nous décrivons cette structuration normalisée en l'illustrant par des exemples tirés des didacticiels produits au cours du projet.

1.1 - Didacticiel

1.1.1 - Définition

Le didacticiel gère l'interaction entre l'apprenant et l'auteur, via le système. "Il est composé de dialogues (cf. §1.2) qui traitent chacun un point pédagogique précis. Il peut être décrit statiquement comme un graphe dont les sommets sont des dialogues entre lesquels existent des liaisons" [ADI 81] (figure 1). Son exécution donne lieu à un parcours particulier de ce graphe, où le nombre et l'ordre des dialogues effectivement exécutés dépendent du comportement de l'apprenant. A chaque dialogue est associée une unité de liaison qui décrit les conditions de passage d'un dialogue à un autre, sur la base des informations de l'environnement didacticiel (cf. §1.1.3) et éventuellement de l'environnement apprenant (cf. §1.1.2).

Un didacticiel peut correspondre à plusieurs heures de travail pour l'apprenant. Aussi, celui-ci travaille-t-il par sessions. On trouvera en annexe un exemple de session de travail avec le didacticiel FLE de soutien à l'enseignement des temps. Au début de chaque session, l'apprenant doit pouvoir reprendre la suite de ce qui a été fait lors de la session précédente. Ceci est possible par la mémorisation des informations concernant son activité, dans l'environnement didacticiel (cf. §1.1.3).



Di représente le nom du dialogue i du didacticiel

ULi représente le nom de l'unité de liaison attachée au dialogue i.

figure 1 : structure d'un didacticiel

Caractéristiques des didacticiels de F.L.E réalisés :

. Le premier a pour objectif pédagogique un soutien à l'enseignement du passé-composé à des étudiants arabophones [SAP 81] [MOS 82]. Il s'exécute sur un micro-ordinateur et fait intervenir des périphériques audio-visuels : projecteur de diapositives et magnétophone à cassettes à accès aléatoire, ainsi que des périphériques conventionnels : écran, clavier alpha-numérique, imprimante.

. Le second didacticiel est un soutien à l'enseignement des temps : passé-composé, présent et futur, tant sur le plan morphologique que syntaxique. Comme pour le premier didacticiel, le public d'apprenants est composé d'étudiants arabophones. Par contre, la configuration matérielle est plus légère ; il n'y a plus de périphérique audio-visuel piloté par l'ordinateur ; seul un magnétophone à cassettes piloté par l'apprenant est utilisé.

Ce didacticiel est formé de trois sous-graphes disjoints dont les parcours par l'apprenant peuvent se faire de façon indépendante (figure 2) : le début d'une session donne lieu au choix du sous-graphe à parcourir.

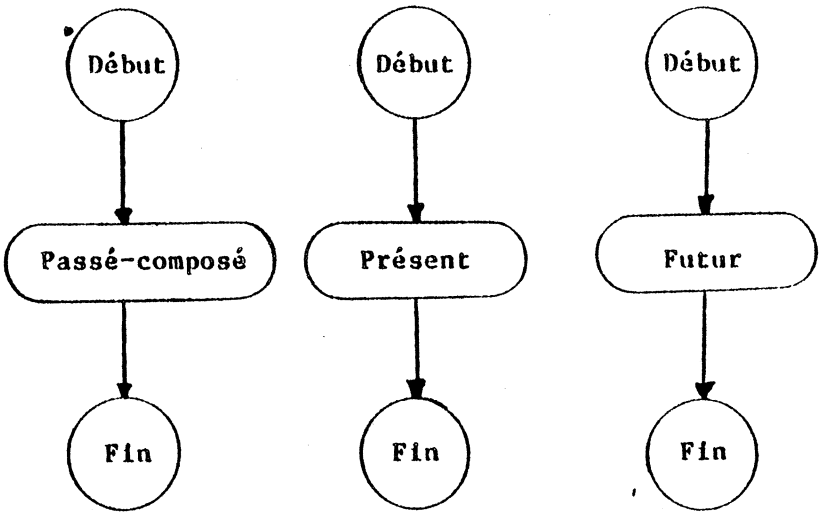


Figure 2 : didacticiel "présent passé-composé futur"

1.1.2 - Environnement Apprenant

On appelle "environnement apprenant" un ensemble d'informations fournies par l'élève et (ou) par l'enseignant ; elles concernent l'identification de l'apprenant. Elles sont, a priori, les mêmes pour tous les didacticiels mis à la disposition de l'apprenant.

Il s'agit de :

- son état civil
- son identification pour le système EAO
- sa classe
- sa scolarité
- etc...

Certaines de ces informations peuvent être utilisées pour personnaliser un dialogue : par exemple, pour afficher des messages au masculin ou au féminin, ou encore des messages faisant apparaître le prénom de l'apprenant. D'autres peuvent faire intervenir des éléments linguistiques (par exemple des expressions locales) liés à la culture de l'apprenant.

1.1.3 - Environnement didacticiel

Dans ce que l'on appelle "environnement didacticiel", on regroupe :

- des informations définies par l'auteur, retenues à la fin de chaque dialogue ; elles sont élaborées par l'unité de liaison associée au dialogue (exemple : nombre d'erreurs, pourcentage des réponses exactes, note) ;
- des informations produites par le "système" (exemple : cheminement de l'apprenant, chronométrages, etc...).

Ces informations permettent :

- de prendre des décisions en fonction de l'activité de l'apprenant, par exemple déterminer le dialogue suivant à exécuter;
- d'évaluer et de valider le didacticiel.

Nous avons divisé l'ensemble de ces informations en deux groupes :

- l'environnement didacticiel apprenant qui regroupe les informations concernant l'activité de l'apprenant;
  (exemple : dernier dialogue effectué par l'apprenant).

- l'environnement didacticiel classe qui regroupe les informations concernant l'activité d'un groupe homogène d'apprenants (classe); il n'est pas modifié au cours d'une session, mais peut l'être par l'enseignant utilisateur au cours d'une session d'évaluation du travail du groupe d'apprenants.
  Dans les didacticiels du projet MOSAIQUE, l'environnement didacticiel classe sera défini lors de l'expérimentation par les enseignants utilisateurs.

1.2 - Dialogue

1.2.1 - Définition

Un dialogue est un ensemble d'interactions appelées situations pédagogiques, entre l'apprenant et l'auteur via le système EAO. Il doit permettre d'atteindre un objectif pédagogique précis à partir d'un niveau minimal déjà acquis par l'élève. Il est autonome et doit pouvoir être utilisé dans des contextes différents par des enseignants (pour la construction d'un nouveau didacticiel par exemple). Une fois conçu et réalisé, c'est un sous-ensemble indivisible du didacticiel : c'est un logiciel fermé, éventuellement adaptable à des besoins spécifiques d'un enseignant utilisateur, si ces adaptations ont été prévues par l'auteur.

1.2.2 - Environnement dialogue

On appelle "environnement dialogue" l'ensemble des informations manipulées dans un dialogue. Des échanges avec l'apprenant, on tire des informations en fonction des indications données par l'auteur,

et d'actions faites automatiquement par le système. Ces informations sont ensuite fournies à l'unité de liaison pour la mise à jour de l'environnement didacticiel et le calcul du dialogue suivant à exécuter.

1.2.3 - Situation pédagogique

Un dialogue est constitué de **situations pédagogiques** entre lesquelles existent des liaisons (cf. figure 3). Une situation pédagogique est l'unité minimale de décomposition d'un dialogue. "C'est l'environnement technico-pédagogique que l'on propose à l'apprenant à un moment donné, en vue d'une réaction de sa part (réaction pouvant aller de la simple observation à une activité plus sophistiquée)" [Sch 83].

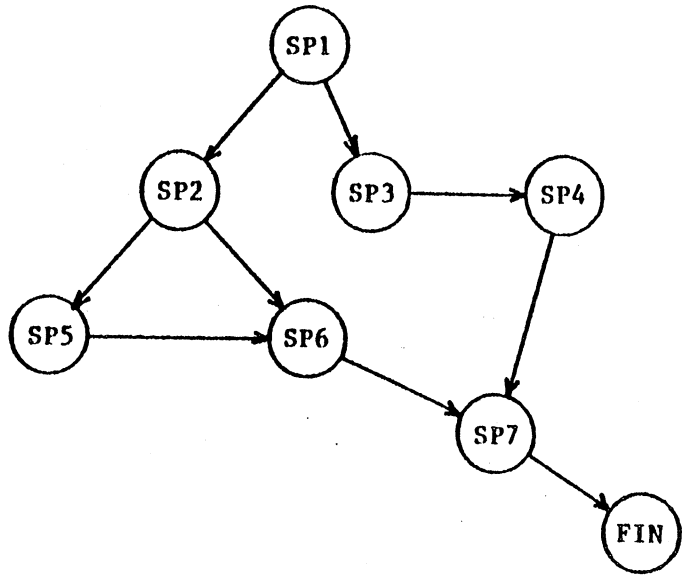
Une situation pédagogique est formée :

- **d'une ou plusieurs actions spécifiées par l'auteur**, qui réalisent des échanges avec l'apprenant, modifient éventuellement l'environnement dialogue et effectuent des opérations particulières sur les objets manipulés (cf. §1.4).

  . la présentation d'une scène
  . la sollicitation d'une réponse de l'apprenant
  . la réception d'une réponse
  . la mise sous une forme canonique d'un message ou d'une réponse
  . l'analyse d'une réponse
  . la réaction à une réponse
  . le suivi de l'activité : conservation d'informations (traces), calculs

- **d'une unité de décision** qui calcule le passage à une autre situation pédagogique en fonction de l'environnement dialogue.

Exemple : La dernière situation pédagogique d'un dialogue consiste en général à présenter à l'apprenant ses résultats : appréciation, score, pourcentage d'erreurs, etc...



SPi représente la situation pédagogique i.

figure 3 : structure d'un dialogue

Les situations pédagogiques utilisées dans les didacticiels F.L.E. sont classées en :

- situation pédagogique de présentation,
- situation pédagogique d'activité de l'apprenant,
- situation pédagogique de corrections,
- situation pédagogique de renforcement.

1.2.4 - Unités de liaison

A chaque dialogue est associée une unité de liaison ; celle-ci s'exécute à la fin du dialogue. Elle a pour rôle :

- de mettre à jour l'environnement didacticiel à partir de l'environnement dialogue,
- de calculer le dialogue suivant à exécuter en fonction des critères choisis par l'auteur.

Le calcul du dialogue suivant peut parfois être assez complexe. Il peut tenir compte de résultats de plusieurs dialogues exécutés auparavant. A l'opposé, il peut aussi dans certains cas être choisi au hasard.

Exemple : Après la détection de certaines erreurs sur un problème particulier, on présente à l'apprenant au début de chaque session, un exercice qui le fait travailler sur le même problème. En fonction de ses résultats, on décide de poursuivre ou non cette activité.

1.3 - Requêtes

Une requête est une demande de l'apprenant d'interrompre le dialogue en cours, en vue :

- d'accéder à des ressources complémentaires comme par exemple un dictionnaire, une calculette ou des informations complémentaires concernant l'exercice en cours,
- d'exécuter un autre dialogue.

L'apprenant peut envoyer sa requête chaque fois qu'il a le contrôle. En cours d'exécution d'une requête, il peut être possible, dans certains cas, d'en appeler une autre. En fin d'exécution, on se retrouve à l'endroit où le dialogue (ou la requête précédente) a été interrompu sauf demande explicite due à la nature de la requête, par exemple la demande d'arrêt du travail en cours.

La fonction de requête est très importante du point de vue pédagogique. Ce sont des outils que l'auteur met à la disposition de l'apprenant pour l'aider dans son travail. Ainsi, elles peuvent permettre un renforcement immédiat ou constituer une source complémentaire d'informations, lorsque l'apprenant rencontre une difficulté. A tout moment, l'auteur a la possibilité d'inhiber (interdire) ou d'activer (autoriser) des requêtes (cf. §1.4).

A titre d'exemple, nous donnons les requêtes définies dans les deux didacticiels du projet MOSAIQUE :

CONSIGNE	: explication détaillée de l'exercice en arabe
LEXIQUE	: signification d'un mot (traduction)
VERBE	: conjugaison d'un verbe
REPETITION	: répétition d'un message oral
ECRIRE	: écriture du message oral
GRAMMAIRE	: rappel de la règle de grammaire attachée à l'exercice
ARRET	: arrêt de la session
DEBUT	: recommencement du dialogue en cours
SECOURS	: aide complémentaire pour résoudre l'exercice (propre à l'exercice)
REPONSE	: donne la bonne réponse attendue
BOITE A LETTRES	: possibilité d'envoyer des messages à l'enseignant.
MEMENTO	: rappel de l'utilisation de certaines touches particulières du clavier

1.4 - Actions élémentaires

Les actions élémentaires forment le niveau le plus bas des interactions apprenant-système ; ce sont les suivantes :

- envoi d'un message informatif ou d'une sollicitation de réponse
- réception d'une réponse de l'apprenant
- mise sous forme canonique d'une réponse
- analyse d'une réponse
- gestion d'informations pédagogiques
- exécution d'un programme externe (par exemple un module de simulation ou l'accès à une base de données).

1.4.1 - Envoi d'un message ou d'une sollicitation

Cette action permet l'envoi à l'apprenant d'une information pouvant être sous la forme :

- d'un texte,
- d'un dessin,
- d'une image fixe ou animée,
- d'un son.

On parle dans ce cas d'information simple. Lorsque plusieurs informations simples sont composées en un seul message informatif, on parle de scène. Une information est envoyée à l'apprenant par un média (périphérique) compatible avec la nature de l'information.

Remarque : Une sollicitation est un message (ou une scène) particulier ayant pour but de demander à l'apprenant d'intervenir en envoyant un ou plusieurs messages au système. Son envoi sera toujours suivi d'une réception de réponse de l'apprenant.

1.4.2 - Réception d'une réponse de l'apprenant

Cette action permet de recevoir une information de l'apprenant. La nature de cette information dépend du média utilisé :

- texte si le clavier est utilisé,
- dessin pour une tablette graphique,
- code pour une touche de fonction, un écran tactile ou un crayon lumineux.

Remarque : C'est pendant la réception d'une réponse que l'apprenant a la possibilité d'appeler une requête. Il faut donc prévoir un mécanisme (convention) permettant de distinguer une réponse d'une requête.

1.4.3 - Mise sous forme canonique

La mise sous forme canonique transforme la réponse de l'apprenant en une forme normalisée, en vue de faciliter l'analyse de cette réponse et éventuellement de tolérer certaines erreurs. Elle permet une certaine liberté d'expression à l'apprenant car une forme fixe ne lui est pas imposée. Elle constitue un filtre appliqué aux réponses reçues et à celles qui sont prévues par l'auteur, afin de pouvoir les comparer.

Une mise sous forme canonique dépend :
- de la nature du message reçu
- de l'analyse que l'on veut ensuite appliquer.

Suivant la nature du message, des informations complémentaires sont nécessaires :

- vocabulaire
- table de transcodage
- grammaire
- etc...

Les informations à mettre sous une forme canonique dans les didacticiels produits au cours du projet sont essentiellement textuelles (phrases en français). Les filtres définis sur ce type d'informations sont les suivants :

- banalisation minuscules/majuscules (transcodification partielle)
- troncature de réponse
- suppression de mots
- construction d'une liste d'éléments de réponse (ensemble de mots identifiés).

Exemple : Banalisation majuscules / minuscules et accents :

Réponse attendue :

Il n'a pas mangé que des fruits mûrs.

Après mise sous forme canonique :

il n'a pas mange que des fruits murs.

Réponse de l'apprenant :

il n'a pas mangé que dés fruits murs.

Après mise sous forme canonique :

il n'a pas mange que des fruits murs.

Après la mise sous forme canonique, les deux réponses sont identiques. L'auteur peut donc admettre la réponse, s'il estime qu'au stade où se trouve l'apprenant, les erreurs d'accent sur "des" et "mûrs", ou l'erreur de majuscule sur "Il" ne sont pas significatives. Sinon, une comparaison avec la réponse prévue accentuée permettra de détecter et de signaler ces erreurs.

1.4.4 - Analyse de réponse

"L'action d'analyse de réponse consiste à identifier s'il y a coïncidence entre la réponse élève et l'une des réponses prévues par l'auteur" [ADI 81] : elle détermine la correction de la réponse et (ou) la nature de l'erreur détectée.

En général, il faut extraire de la réponse les informations pertinentes spécifiées par l'auteur, puis en vérifier l'exactitude. Des informations complémentaires calculées lors de la mise sous forme canonique peuvent également être utilisées.

Dans les didacticiels F.L.E., les analyses de réponse définies sont :

- la comparaison terme à terme,
- vérification de l'absence ou de la présence de mots-clés,
- vérification de l'ordre des mots d'une phrase,
- reconnaissance du radical et de la terminaison d'un verbe.

La qualité de l'analyse de réponse est un critère important de jugement d'un didacticiel. C'est souvent une des parties complexes d'un dialogue. En effet, si l'on fait abstraction des questionnaires à choix multiples (QCM) dont l'analyse de réponse est triviale, d'autres approches peuvent mettre en oeuvre des techniques de calcul puissantes [Mar 83], et parfois complexes : analyse syntaxique, recherche de mots clés, reconnaissance de mots avec tolérance orthographique, simulations, etc...

Exemple : Au cours de la phase de formation ou début du projet (cf. Introduction)., fut réalisée une maquette de didacticiel faisant intervenir une analyse de réponse formulée à l'aide d'une grammaire LL(1).

1.4.5 - Gestion des informations

Les actions de gestion des informations permettent :

- **la modification d'informations.** Il s'agit essentiellement de :

  . la mise à jour des environnements dialogue et didacticiel,

  . l'inhibition ou l'activation de requêtes;

- **la récupération d'informations** (traces) au cours de l'exécution du didacticiel. Elles n'interviennent pas sur le déroulement, mais contribuent à une évaluation de l'activité de l'apprenant et à une validation du didacticiel (exemple : trace des erreurs de grammaire).

Dans les didacticiels produits, on distingue les niveaux de traces suivants :

- la réponse,
- le dialogue,
- la catégorie d'exercice,
- la session.

1.4.6 - Exécution d'un programme externe

Certains programmes sont conçus et réalisés indépendamment du didacticiel (par exemple, des modules de simulation ou un logiciel graphique). Ils doivent pouvoir être facilement utilisés depuis un dialogue. L'utilisation de tels programmes apparaît sous la forme d'actions élémentaires spécifiques, au niveau du dialogue. Ceci nécessite une forme standard de communication entre un dialogue et tout logiciel externe.

1.5 - Conclusion

Cette structuration d'un didacticiel a semblé bien adaptée, tant à la conception par l'auteur qu'à la réalisation informatique et à l'implantation sur micro-ordinateurs.

a) Sur le plan pédagogique :

La notion de dialogue semble familière aux auteurs qui la connaissent dans l'enseignement au travers de notions telles que chapitre de livre, exercices, etc...

On peut opposer cette manière de faire à la suivante :
Un didacticiel peut être considéré comme un tout indissociable traitant d'un sujet global. Il n'y a pas de notion de dialogue, ce qui peut éventuellement s'adapter à une certaine forme de travail pédagogique. Une suite de scènes est présentée à l'apprenant jusqu'à ce que l'objectif pédagogique soit atteint.

Cette non structuration présente l'inconvénient de poser des difficultés à un enseignant utilisateur, pour réaliser les adaptations nécessaires à ses besoins, notamment la construction d'un autre didacticiel à partir de celui existant. D'autre part, au niveau de la réalisation, un découpage devra être fait en raison des ressources limitées des matériels. Celui-ci sera plus complexe à définir si des parties indépendantes ne sont pas mises en évidence.

b) Sur le plan informatique :

Cette décomposition d'objets en sous-objets indépendants ne peut que faciliter la réalisation (modules séparés) et l'installation sur des matériels à ressources limitées (recouvrement facile à concevoir).
De plus, une autre structuration est envisageable : elle consiste à considérer le didacticiel comme un ensemble de processus gérés par un système. Les processus sont créés, interrompus ou détruits en fonction des réactions des apprenants. Les processus actifs sont les dialogues et requêtes en cours d'exécution à un instant donné.
Cette structuration s'adapte aux environnements multi-utilisateurs. Du point de vue pédagogique, la spécification est la même que dans la structuration en modules, un dialogue étant l'équivalent d'un processus.

CHAPITRE II

SPECIFICATION D'UN DIDACTICIEL PAR UN AUTEUR

2.1 - Introduction

De l'expérience MOSAIQUE, nous avons tiré un modèle de la spécification d'un didacticiel par un auteur, que nous présentons ici. L'application de ce modèle est ensuite illustrée dans le contexte des didacticiels produits dans le projet.
La démarche de réalisation d'un didacticiel peut se décomposer en cinq phases (cf. aussi [Lan 83]) :

1 - L'analyse pédagogique qui a pour but de définir le produit, c'est à dire :

- l'objet du didacticiel : son contenu (sujet, niveau, connaissances prérequises, etc...), son rôle dans le processus d'apprentissage par rapport aux autres ressources pédagogiques (soutien, apprentissage, contrôle de connaissances, etc...),
- la population des apprenants à qui s'adresse le didacticiel : leur origine, leur niveau, les connaissances prérequises, etc...
- les approches mises en oeuvre, c'est à dire la démarche d'apprentissage dans la discipline enseignée, qui va servir de base dans la structuration en dialogues,
- les médias mis en oeuvre, dans le didacticiel,
- les ressources mises à la disposition de l'apprenant au travers des requêtes,
- les modalités d'utilisation : lieu de travail, rythme, exercices faits seul ou à plusieurs, etc...

2 - L'élaboration du scénario où l'auteur spécifie complètement le didacticiel. Il décompose la matière à enseigner en points pédagogiques précis, et définit les dialogues correspondants. Pour chaque dialogue, il spécifie : les scènes présentées, les réponses attendues, l'analyse des réponses, les réactions en fonction des erreurs, les informations à conserver et les requêtes disponibles. Enfin, il définit les cheminements possibles dans le didacticiel.

3 - La réalisation informatique qui consiste à passer des spécifications des auteurs à un programme informatique. Cette phase est effectuée soit par l'auteur [Val 73], à l'aide d'outils spécialisés (langages auteurs [ChS 80] RoR 81] [Dav 83], systèmes EAO [Log 79] [CDF 81] [Kea 82] [ADI83])), soit par un informaticien ; dans ce cas, l'auteur est libéré des problèmes informatiques, mais une étroite collaboration avec l'informaticien est nécessaire. Cette dernière démarche est celle que nous avons choisie. Les choix faits pour la réalisation (méthodes de programmation mises en oeuvre, langage de programmation utilisé, configuration de la machine cible) sont de grande importance : le didacticiel produit doit être fiable et facilement transportable sur un autre site.

4 - L'expérimentation qui permet de vérifier l'adéquation du didacticiel aux objectifs pédagogiques. Elle consiste à soumettre à des apprenants le didacticiel réalisé. Elle entraîne en général des modifications qui améliorent la qualité pédagogique du didacticiel.

5 - L'industrialisation qui pose le problème du support de diffusion, du suivi et de l'adaptabilité du didacticiel à des besoins propres d'enseignants utilisateurs.

L'analyse pédagogique (la première phase) est spécifique du domaine à enseigner ; nous ne la traitons pas ici, nous renvoyons le lecteur aux documents pédagogiques du projet [SAP 81] [Sch 83].
Une fois cette analyse pédagogique faite, l'auteur doit spécifier le contenu du didacticiel. Nous nous intéressons ici essentiellement à cette phase d'élaboration du scénario.

Pour pouvoir spécifier le scénario, l'auteur doit avoir une certaine connaissance des possibilités de l'ordinateur et du système mis à sa disposition :

- sur le plan de la communication homme-machine pour spécifier les scènes et les solicitations,

- sur le plan du traitement des informations, notamment pour spécifier les analyses de réponses et les réactions du système :

  . quelles sont les erreurs à détecter, quelles sont les méthodes et les règles de reconnaissance de ces erreurs,

  . quels sont les diagnostics envoyés, quelles erreurs sont tolérées, signalées ou non, quelles informations sont conservées en vue d'une validation pédagogique du dialogue et du calcul du dialogue suivant à présenter (pour assurer le cheminement de l'apprenant dans le didacticiel).

Plus l'auteur est familier avec l'ordinateur, meilleure est l'utilisation de ses possibilités. Cette connaissance peut se faire progressivement et s'améliorer, éventuellement aidée par l'informaticien au cours du projet. Toutefois, la spécification doit être faite le plus possible en toute indépendance des solutions informatiques.

2.2 - Les problèmes de spécification du scénario

Pour chacune des phases de l'élaboration du scénario (cf §2.1) nous décrivons les problèmes de spécification qui se posent à l'auteur.

2.2.1 - Décomposition du didacticiel en dialogues

D'après ce qui a été dit, cette phase est purement pédagogique : l'auteur définit chaque dialogue par :

- son objectif pédagogique précis, par exemple : reconnaissance des terminaisons du participe passé (i,u,é),
- l'activité demandée à l'apprenant, exemples : relier des éléments, classer, discriminer
- l'approche adoptée, c'est à dire la manière de faire travailler l'apprenant, exemples : reconnaissance orale, reconnaissance écrite, production créatrice, etc.
- la durée de l'exercice

Les dialogues ainsi définis, l'auteur peut définir leur contenu.

2.2.2. - Spécification de la présentation de scènes et de sollicitations

En général, plusieurs médias interviennent dans la présentation d'une scène. L'auteur doit spécifier :

- **les informations** présentées par chaque média concerné. Pour cela on a besoin d'une forme de description des informations envoyées, adaptée à chaque média.
- **l'ordre** dans lequel ces informations sont envoyées, et éventuellement la durée de leur présentation. Un formalisme permettant d'exprimer la présentation d'informations et la synchronisation (séquencement, simultanéité, pauses) s'avère donc nécessaire.

Afin de faciliter la manipulation et la description des informations présentées, il faut un moyen permettant de nommer ces informations.

Remarque : Une sollicitation est une scène dont la fonction est d'inciter l'apprenant à intervenir. Elle sera toujours suivie d'une réception d'informations de l'apprenant. Sa spécification est identique à celle d'une scène.

2.2.3. - Spécification de la réception d'informations

Pour spécifier une réception d'informations, l'auteur doit définir :

- le **média de réception** : clavier alpha-numérique, clavier numérique, touches de fonctions, écran tactile, etc...
- la **nature de la réponse** attendue, en vue de vérifier sa validité, de lui appliquer une analyse et éventuellement de gérer l'écho à l'écran : phrase alphabétique, valeur numérique, formule, code, etc...

Remarque : A chaque nature de réponse possible, on peut définir un alphabet des caractères admis et un protocole de réception, s'ils ne sont pas standards.
Exemple : Réception d'une équation chimique.

- la zone écran où se fait l'écho de la réponse (en cas d'écho). Cette indication permet de limiter la taille de la réponse et de protéger le reste de l'écran.

Au niveau de la réception, l'apprenant peut être amené à demander l'accès à des ressources complémentaires au moyen de requêtes. L'auteur doit définir un protocole d'accès à ces ressources : touches de fonctions, touche spéciale à frapper avant le nom de la requête, etc.

2.2.4. - Spécification de l'analyse de la réponse

L'auteur doit définir l'ensemble des réponses sous la forme de listes exhaustives ou d'un modèle décrivant une classe de réponses comme par exemple une grammaire. Il doit décrire les erreurs ou classes d'erreurs à reconnaître.

C'est la partie de la spécification qui pose généralement le plus de problèmes à l'auteur. Il aura plus ou moins de difficultés :
- à prévoir et à décrire tous les cas possibles de réponses, à les exprimer sous la forme d'un modèle,
- à définir la stratégie d'analyse des réponses, c'est à dire à choisir parmi les techniques de comparaison à sa disposition, celles à appliquer à la réponse. L'auteur doit définir :
  - les filtres (mises sous forme canonique) à appliquer pour tolérer certaines erreurs ou accepter des formes différentes d'une même réponse, avant la comparaison,
  - les techniques de comparaison à appliquer successivement,
  - l'ordre dans lequel ces filtres et techniques de comparaison sont à appliquer ; il correspond en général à l'ordre d'importance des erreurs détectées.

Dans certains systèmes de production de didacticiels, l'auteur non informaticien a à sa disposition un ensemble d'opérations d'analyse prédéfinies directement applicables aux réponses reçues. Ces opérations peuvent être élaborées par l'auteur, avec la collaboration d'un informaticien, au cours de la spécification du didacticiel.

Exemple : Pour une réponse donnée sous la forme d'une phrase en français, appliquer successivement :

- le filtre de banalisation majuscules/minuscules,
- la recherche des mots clés de la réponse (vérification de la présence de tous les mots clés attendus),
- la vérification de l'ordre des mots-clés,
- la recherche des erreurs de majuscule.

L'oubli d'un mot-clé peut être défini comme plus grave qu'une erreur d'ordre qui elle-même peut être considérée comme plus grave qu'une erreur de majuscule. Si l'oubli d'un mot est détecté, l'auteur choisit d'arrêter immédiatement l'analyse ou de rechercher d'autres erreurs.
Les erreurs sont cherchées en fonction de leur hiérarchie. Suivant l'importance de la première erreur détectée, les autres seront plus ou moins significatives. Il n'est donc pas toujours nécessaire de rechercher toutes les erreurs d'une réponse donnée.

La spécification de l'analyse d'une réponse consiste donc pour l'auteur à définir :
- l'ensemble des réponses attendues par un formalisme adapté,
- les opérations d'analyse à appliquer et leur ordre.

2.2.5. - Spécification des réactions aux réponses après analyse

L'auteur doit spécifier une scène correspondant à la réaction à la réponse reçue de l'apprenant. Cette scène dépend du résultat de l'analyse. Il faut spécifier
- les scènes possibles à présenter,
- les conditions de présentation de chaque scène.

Certaines des conditions à spécifier ne sont pas exclusives, c'est le cas lorsque plusieurs erreurs de types différents sont détectées. Il faut définir la priorité de certaines conditions par rapport à d'autres.

Exemple : Dans un exercice sur la structure d'une phrase, une erreur d'ordre sur les mots peut être considérée comme étant plus grave qu'une erreur d'orthographe.
L'auteur spécifie le traitement de l'erreur en fonction de son importance.
La scène correspondant à l'erreur la plus grave sera présentée à l'apprenant. Celle-ci sera suivie d'une nouvelle sollicitation pour demander une correction de la réponse.

Remarque : Ce schéma "demande de correction, réception, réanalyse, diagnostic", est classique en EAO et peut être réitéré. L'auteur spécifie la condition d'arrêt de l'itération et les opérations à faire à la sortie, comme par exemple la présentation de la réponse correcte attendue.

2.2.6. - Spécification du "suivi" de l'activité de l'apprenant

L'auteur doit spécifier les informations qu'il veut obtenir à la fin de chaque dialogue :

- vis à vis de l'apprenant, pour une évaluation de son activité dans le dialogue, et déterminer l'exercice suivant.
- vis à vis du didacticiel, pour une validation pédagogique de celui-ci.

Ces informations sont contenues dans l'environnement dialogue (cf. §1.2.2) et dans les traces.
Il est donc nécessaire d'avoir un formalisme pour exprimer :

- Les informations à conserver sur fichiers (traces) après l'analyse de réponse,
- les calculs à effectuer au cours de l'exécution du dialogue : comptage des erreurs, des sollicitations, chronométrages, nombre d'appels aux requêtes, etc. A chacun de ces résultats l'auteur donne un nom en vue de faciliter la spécification d'autres calculs.

L'expérience montre qu'un certain nombre de résultats sont à calculer pour **tous** les dialogues d'un didacticiel.

Exemple : Nombre d'appels de chaque requête, pourcentage d'erreurs.

L'auteur spécifie ces informations comme étant l'environnement calculé **par défaut**. Ces calculs sont effectués pour tous les dialogues ; l'auteur n'a pas à les respécifier à chaque fois. Seuls les résultats particuliers à calculer en plus de l'environnement par défaut, sont à spécifier pour un dialogue donné.

2.2.7 - Spécification des requêtes

Dans chaque dialogue, un certain nombre de requêtes sont mises à la disposition de l'apprenant. L'appel d'une requête est fait par l'élève, au moment où celui-ci a le contrôle, lors de la réponse à une sollicitation.

L'exécution d'une requête est identique à celle d'un dialogue ; seul l'objet de l'exécution change : accès à une ressource au lieu du traitement d'un point pédagogique. La spécification d'une requête est donc en partie la même que celle d'un dialogue : scènes et sollicitations présentées, réception de réponses, analyses, réactions aux réponses qui consistent en général en la présentation des informations demandées par l'apprenant.

Catégories de requêtes :

Nous distinguons trois types de requêtes :

- les **requêtes globales**, communes à tous le didacticiel comme par exemple l'accès à la conjugaison des verbes du français.

- **les requêtes globales particulières,** appelées de façon identique à partir de tous les dialogues, mais dont le contenu diffère (scènes différentes d'un exercice à l'autre) ; c'est le cas par exemple de la requête d'accès à la consigne de l'exercice courant.

- **les requêtes locales,** dont l'appel et le contenu sont spécifiques à un dialogue donné.

Certaines requêtes risquent de modifier l'état des médias au moment de l'interruption du dialogue. Il faut prévoir pour ce type de requêtes une mémorisation de l'état des médias qu'elles modifient.

Remarque : Certaines requêtes ont une fonction d'interruption de l'activité en cours ; c'est le cas par exemple de la requête de demande d'arrêt de la session en cours ou de la demande d'interruption du dialogue en cours pour le recommencer ou en exécuter un autre. Pour ces requêtes l'auteur n'a à spécifier que la fonctionnalité de la requête, son mode d'appel et la scène à présenter s'il y a lieu en cas d'appel.

2.2.8. - Spécification des cheminements possibles de l'apprenant dans le didacticiel

Lorsque les dialogues forment une certaine progression dans le didacticiel, l'auteur spécifie les enchaînements possibles des dialogues.

Un formalisme est nécessaire pour spécifier ces enchaînements, et les conditions de passage d'un dialogue à un autre. La présentation des enchaînements sous la forme d'un graphe convient lorsque tout dialogue se place dans la progression définie par l'auteur. Elle est insuffisante et trop rigide lorsque les dialogues sont indépendants les uns des autres, ou regroupés par niveaux de progression.

Exemple :

Les dialogues sont classés en sous-ensembles ; tant que la moitié des exercices d'un niveau donné n'est pas faite, l'apprenant ne peut passer au niveau suivant. Le choix d'un exercice dans un niveau donné peut se faire entre autres :
- par l'apprenant qui choisit dans un menu
- au hasard par le système
- par le système, selon des critères donnés par l'enseignant.

Afin d'offrir une plus grande souplesse d'utilisation de son didacticiel, l'auteur peut donner à l'enseignant utilisateur la possibilité de modifier ou de redéfinir les cheminements possibles qu'il a prévus.

2.3 - Résolution des problèmes de spécification dans le projet MOSAIQUE

Nous illustrons ici l'application du modèle de spécification présenté, dans le contexte des didacticiels produits au cours du projet.

2.3.1. - Objectifs

Caractéristiques du produit à réaliser :

Nous devions produire un logiciel de soutien dans l'enseignement du Français Langue Etrangère, plus particulièrement l'enseignement des trois temps fondamentaux du discours : passé-composé, présent, et du futur. Le public visé est composé d'étudiants de troisième cycle, de culture arabe et considérés comme "faux débutants" en français

La notion d'approche est l'un des points forts de la spécification pédagogique et de la méthode pour structurer le didacticiel en dialogues [Sch 83].
Les approches mises en oeuvre dans les didacticiels FLE sont diversifiées : reconnaissance orale, écrite, réflexion, production créatrice, production imitative [SAP 81].

Les médias intervenant dans les didacticiels et pilotés par le micro-ordinateur sont :
- un clavier alpha-numérique,
- un écran vidéo noir et blanc,
- un magnétophone à cassettes à accès aléatoire (version 1),
- un projecteur de diapositives (version 1),
- une imprimante.

Les ressources mises à la disposition de l'apprenant au travers des requêtes sont essentiellement :
- un lexique français-arabe,
- l'accès à la conjugaison de tous les verbes apparaissant dans le didacticiel :
- des compléments concernant les exercices proposés : consigne de l'exercice en arabe (magnétophone), rappel de règles de grammaire, explications complémentaires.

Le didacticiel produit doit de plus être facilement transportable sur de nombreux micro-ordinateurs.

Moyens définis pour la réalisation :

De manière classique, les auteurs disposent d'un environnement auteur, en général un langage, et plus recemment des outils plus évolués (environnements). Ceci nécessite du temps pour la formation et peut influencer les auteurs. Nous avons choisi des conditions expérimentales faisant intervenir une équipe à double compétence (cf.introduction), ce qui a permis de libérer les auteurs des contraintes informatiques.

2.3.2. - La présentation de scènes

Les médias intervenant dans le didacticiel sont diversifiés. Pour décrire les informations à présenter, un certain nombre de fiches adaptées à la nature des informations sont utilisées par les auteurs. Lorsque toutes ces informations sont définies, l'auteur décrit **l'algorithme de présentation** de la scène : ordre d'affichage des informations, pauses éventuelles, etc...

- **Les informations présentées à l'écran** sont, soit textuelles, soit formées de caractères semi-graphiques (cadres, tableaux, dessins). Sur une grille aux dimensions de l'écran, l'auteur décrit les zones d'affichage des informations et les zones de réception des réponses. Il les désigne par un numéro qui sert à les désigner dans la description de l'algorithme.
  Séparément, chaque information présentée à l'écran est décrite par son contenu et son nom.

- **Les informations sonores** sont enregistrées sur bandes magnétiques. Elles sont décrites par le texte du message oral, s'il s'agit d'une information textuelle, ou par une description du message sonore. A chaque message est associé un **nom** qui sert à la description des algorithmes de présentation de scènes.

- **Les diapositives** sont définies comme les messages sonores, par un nom et un texte décrivant leur contenu.

L'algorithme de présentation de la scène est écrit de façon informelle en français. Cependant un certain nombre de symboles ou mots-clés sont définis pour décrire les opérations de base :
- affichage d'un message à un point d'ancrage de l'écran,
- pause,
- présentation d'un message sonore,
- présentation d'une diapositive,
- etc...

Toute description présentant un caractère ambigu ou difficile à exprimer est spécifiée en français.

2.3.3. - La réception d'informations

Dans la configuration utilisée, un seul média permet la réception de réponses : le clavier alpha-numérique.
Deux modes de réception sont définis :

- la réception d'une réponse dans la zone écran définie par l'auteur, l'écho est contrôlé au fur et à mesure de la réception.
- la modification d'une réponse déjà donnée, à l'aide de touches de fonctions spécifiques.

Dans les deux modes, l'apprenant a la possibilité d'appeler des requêtes en utilisant une touche spéciale avant le nom de la requête. Ce mode d'accès à une requête a été défini par les auteurs ; il est lié au didacticiel. Un symbole est défini pour décrire chaque mode de réception.

La description d'une réception d'information, dans l'algorithme de déroulement du dialogue, consiste donc pour l'auteur à donner :
- le symbole correspondant au mode de réception choisi,
- le nom de la réponse attendue,
- la zone écran dans laquelle est fait l'écho, s'il y a écho.

2.3.4. - La mise sous forme canonique et l'analyse de réponse

En collaboration avec les informaticiens, les auteurs définissent un certain nombre de schémas d'analyse de réponse, qui font intervenir des opérations de base :

. entre les réponses reçues et les réponses attendues :

- la comparaison terme à terme,
- la vérification de présence de certains mots,
- la vérification de l'ordre des mots dans une phrase,
- etc...

. directement sur la réponse reçue avant comparaison :

- filtre orthographique, c'est à dire reconnaissance d'un mot à une lettre près,
- filtre sur la ponctuation,
- filtre sur les accents,
- filtre sur les lettres majuscules.

Des symboles (mots-clés) sont définis pour représenter chaque schéma d'analyse. L'auteur choisit l'analyse qu'il souhaite appliquer en donnant le nom du schéma prédéfini correspondant.

Cette démarche a été choisie dans les buts de définir des schémas d'analyse adaptés aux besoins des auteurs, et d'en extraire les opérateurs de base pour la définition d'autres schémas d'analyse.

2.3.5. - Les réactions aux réponses

Les réactions aux réponses sont des scènes représentées en fonction des erreurs détectées lors de l'analyse. Soit elles s'intègrent dans les schémas d'analyse prédéfinis, soit elles apparaissent explicitement dans la description des auteurs.

2.3.6. - Le "suivi" de l'activité de l'apprenant

Les calculs à effectuer au cours de l'exécution du dialogue sont prédéfinis et intégrés dans les schémas de base. Une trace de tout ce qui est fait par l'apprenant est construite. Certaines erreurs spécifiques sont à conserver dans un fichier ; un bordereau est défini pour décrire le type de l'erreur à conserver et le nom de la réponse erronée à conserver.

2.3.7. - Les requêtes

Toutes les requêtes définies dans le didacticiel sont des requêtes globales : elles sont appelées de façon identique depuis tous les dialogues. Seul le contenu de certaines scènes diffère d'un dialogue à l'autre. C'est le cas par exemple de la requête qui présente la consigne de l'exercice en cours.

Les actions d'inhibition et d'activation de requêtes sont décrites dans l'algorithme de déroulement du dialogue donné par l'auteur. Il y a une certaine cohérence ou logique pédagogique des auteurs dans le choix d'autoriser ou d'interdire certaines requêtes. Par exemple, dans les didaciticiels produits, les requêtes "verbe", "lexique", "arrêt", et "début" (cf. §1.3) sont toujours disponibles; les requêtes "réponse" et "secours" ne le sont jamais à la première saisie de réponse.

2.3.8. - Les cheminements possibles dans le didacticiel

Deux modes de travail sont proposés à l'apprenant :

- un mode où il choisit le dialogue qu'il souhaite exécuter,
- un autre mode où il suit le cheminement défini par l'auteur. La condition de passage d'un exercice à un autre est définie en fonction des résultats d'erreurs obtenus à certains exercices précédents. L'auteur fait parfois intervenir le hasard dans cette description.

Certaines conditions qui se retrouvent assez fréquemment dans le didacticiel, sont **prédéfinies**. L'auteur ne les décrit pas explicitement, mais spécifie simplement le nom de la condition prédéfinie qu'il souhaite appliquer.

2.4 - Conclusion

Notre méthode de travail a permis aux auteurs non-informaticiens, de résoudre leurs problèmes de spécification dans le projet MOSAIQUE. Elle a demandé une forte interaction entre l'équipe informatique et l'équipe pédagogique [MOS 83], et a permis aux auteurs de prendre conscience des problèmes de description, notamment la nécessité d'une **grande précision**, afin d'éviter les ambiguïtés. La formation initiale au projet [Pai 81] a permis aux auteurs cette prise de conscience, et leur a donné une première vision des possibilités de l'ordinateur.

Au cours du projet MOSAIQUE, la coopération pédagogues informaticiens a toujours été privilégiée. Elle a permis de mettre en évidence un ensemble d'objets et d'opérateurs de base associés, nécessaires à la description de didacticiels. Ces opérateurs permettent la définition de **primitives de manipulation** de ces objets, au niveau de la réalisation du programme informatique.

Cette coopération a également permis l'observation du travail des pédagogues. Celle-ci a favorisé la connaissance de leurs besoins et la définition de méthodes de travail adaptées.

CHAPITRE III

PASSAGE DES SPECIFICATIONS DE L'AUTEUR A UN PROGRAMME INFORMATIQUE

Nous venons de présenter les problèmes de spécification de didacticiel posés à un auteur. Leur résolution dans le projet MOSAIQUE a conduit à la définition d'un dossier de spécification de dialogue, constitué par l'auteur.
Nous présentons dans cette partie les méthodes mises en oeuvre dans le projet MOSAIQUE pour produire le programme informatique correspondant à ces spécifications.

3.1. - Qualités attendues des programmes à produire

Les didacticiels à produire doivent présenter les qualités suivantes :

- une grande fiabilité et une bonne robustesse, c'est à dire que les causes d'arrêt accidentel ou de mauvais déroulement du programme doivent être totalement absentes, même en cas de mauvaise utilisation par l'apprenant. En effet, les utilisateurs de ces programmes ne sont pas des informaticiens et sont souvent incapables de détecter les sources d'erreur.
  D'autre part, un didacticiel est un programme qui a un nombre d'utilisateurs potentiels important (auteur, apprenants, enseignants utilisateurs) ; ceci renforce le besoin de fiabilité.

  Tous les cas d'erreur doivent donc être prévus, y compris ceux dûs à une mauvaise utilisation du programme.

- Une grande facilité de modification (modifiabilité) :
  Pendant les périodes de test et d'expérimentation, le didacticiel est sujet à de nombreuses modifications : un dialogue ne correspond pas toujours à ce qu'en attend l'auteur ; d'autre part, lors de l'expérimentation, les réactions de l'apprenant ne sont pas toujours celles que l'auteur prévoyait. Le didacticiel ne devient un produit figé qu'après un temps important

d'expérimentation (car l'enseignement est un domaine mal modélisé). Les programmes doivent donc être structurés de façon à faciliter les modifications demandées par l'auteur. La connaissance précise des types de modifications demandées le plus fréquemment pourra guider cette structuration.

- Une bonne lisibilité, c'est à dire une grande facilité de lecture et de compréhension des programmes. Ceci améliore la modifiabilité des logiciels produits.

- Une portabilité aisée sur de nombreux matériels :
le didacticiel produit est destiné à être diffusé ; il ne doit pas être tributaire d'un matériel particulier. Le choix du langage d'écriture et d'une structure des programmes séparant les parties spécifiques au matériel et au logiciel utilisé, sont des facteurs importants de portabilité.

- Une adaptation aux contraintes physiques et logiques des micro-ordinateurs :
les postes de travail des apprenants, destinés à supporter les didacticiels produits dans le cadre du projet MOSAIQUE sont des micro-ordinateurs. Les programmes produits doivent s'exécuter sur ce type de matériel. Il faut donc tenir compte des contraintes en place mémoire et en mémoire de masse. D'autre part, le choix du langage d'écriture des programmes sera lié à sa disponibilité sur micro-ordinateur.

- Une conformité aux spécifications de l'auteur :
les programmes produits doivent correspondre aux spécifications de l'auteur ; celles-ci doivent comporter le moins possible d'ambiguïtés, ce qui implique une forte interaction entre l'auteur et l'informaticien au cours de la programmation et des tests.

3.2 - Démarche proposée

Le logiciel d'enseignement produit au cours du projet, qui représente une trentaine d'heures de cours, correspond à une masse de programmes très importante. Aussi a-t-il été nécessaire d'appliquer des méthodes systématiques pour sa production. Le projet MEFIA (MEthodologie et FIAbilité) de l'expérience duquel nous bénéficions, a donné lieu à l'étude de méthodes de programmmation ([LSV 77], [Sch 77.b], [Sch 78], [Sch 79]) et à la définition d'une notation algorithmique (MEFIA cf. [CGS 78], [SGC 78], [Cun 79]) qui facilitent la description et améliorent la fiabilité des algorithmes (cf. §3.3). Nous nous sommes proposés de mettre en oeuvre en vue de les valider, ces méthodes de programmation, et ainsi participer au thème général de méthodologie de la programmation.

Les didacticiels ont été décrits dans la notation algorithmique MEFIA. Ce choix a permis d'avoir une forme de description unique, indépendante d'un matériel ou d'un langage de programmation. Il a permis aussi l'obtention de programmes lisibles et fiables grâce aux caractéristiques de la notation.

Pour faciliter cette étape de programmation, un ensemble de règles de description dans la notation MEFIA, des objets à réaliser, ont été définies [Adm 81]. Ces règles tiennent compte des types de modifications possibles de chaque objet, ce qui améliore la modifiabilité des programmes.
Le passage à un langage de programmation disponible sur micro-ordinateur a été fait en appliquant des règles de traduction systématique de la notation algorithmique vers un noyau portable de ce langage. Ceci a permis d'améliorer la fiabilité des programmes, chaque schéma algorithmique étant toujours traduit de la même manière.

Après une brève présentation de la notation MEFIA, nous décrivons les règles de description d'un didacticiel dans cette notation ; puis nous présentons le passage systématique vers le langage de programmation choisi. Enfin nous rendons compte de l'expérimentation de ces méthodes et des possibilités d'automatisation de ce processus.

3.3 - La notation algorithmique MEFIA

3.3.1 - Présentation

MEFIA est une notation algorithmique qui a été conçue dans le but d'offrir aux concepteurs d'algorithmes :

- la possibilité d'exprimer des abstractions mises en évidence au cours de l'analyse (informations nommées, actions nommées) [Mor 83],
- des facilités d'expression des algorithmes (schémas évolués de conception de programmes)
- l'indépendance par rapport à un langage de programmation
- des facilités de spécification des abstractions manipulées.

Sa syntaxe et ses caractéristiques sont présentées dans [SCM 80] et [Mor 83].

La méthode de conception des programmes proposée consiste :

- en une analyse descendante du problème [Wir 71.b] et l'expression des différents niveaux d'abstraction rencontrés au cours de cette analyse,
- en une spécification, par des commentaires placés de façon pertinente dans le programme
  - . des choix faits lors de l'analyse du problème et des raisons de ces choix
  - . de l'ensemble des informations liées aux actions décrites, notamment leurs états initiaux et finals
  - . du rôle logique des variables et des actions intervenant dans la description de l'algorithme.

3.3.2 - Caractéristiques principales

Les principales caractéristiques de la notation MEFIA sont

- le concept d'information nommée :
  c'est un outil qui permet de décrire une information et de s'y référer en faisant abstraction d'une partie de ses caractéristiques.

- le concept d'action nommée :
  c'est ce qui permet de décrire la solution algorithmique d'un problème et de s'y référer en faisant abstraction du processus engendré lors de son exécution.

- la notion d'univers :
  c'est le regroupement en une même entité (l'univers) des informations nommées et des actions nommées qui les manipulent.

- la notion de type nommé :
  c'est un outil permettant de faire abstraction de la représentation d'une information nommée lors de sa spécification.

- la notion d'intervalle nommé :
  cette notion est associée à la gestion d'une table en algorithmique ; elle sert à spécifier les valeurs possibles des indicatifs de la table et la relation entre une table et les indices qui permettent de la gérer.

- la notion de propriété :
  c'est un outil permettant de spécifier, pour une information nommée donnée, les propriétés que doit respecter sa valeur.

Ces outils sont complétés par le "prologue" qui permet la définition de constantes d'exécution.

Ces caractéristiques augmentent, par rapport à un langage de programmation classique, les possibilités de vérification et d'abstraction. D'autre part, elles permettent d'appliquer systématiquement des méthodes de programmation [Sch 79], indépendamment de contraintes logicielles (langage de programmation) ou matérielles.

C'est à partir de cette notation que nous avons défini des règles de description "informatiques" des objets qui constituent un didacticiel.

3.4 - Règles de description d'un didacticiel dans la notation MEFIA

Les règles de description d'un didacticiel dans la notation MEFIA sont décrites dans [Adm 81]. Nous donnons ici les règles s'appliquant aux principaux objets. Pour chaque objet étudié, nous présentons les modifications à effectuer le plus fréquemment.

3.4.0 - Notations

Nous présentons ici les seuls éléments de la notation MEFIA utilisés dans ce chapitre pour décrire les règles de description des objets pédagogiques, (cf.[Sch 79] et [SCM 80]).

3.4.0.1 - Génération de valeurs

a - **Littéraux** : on utilise les conventions habituelles pour dénoter des valeurs de type entier, réel, logique, et caractères.

b - **Expressions** : on utilise le formalisme classique d'expression avec les conventions habituelles relatives aux priorités.

3.4.0.2 - Procédures

Le mécanisme de procédure suit les règles suivantes :

a - Un paramètre ne peut être modifié par la procédure (apparaître en partie gauche d'une affectation).

b - On distingue :

. les actions :

Leur effet est caractérisé par la modification de variables de leur contexte.

action nom (<liste de paramètres>)
<corps de l'action>
faction

Les actions sont utilisées partout où l'on peut avoir une instruction ce qui correspond à une instruction d'appel de procédure classique.

. les fonctions :

Elles calculent une valeur ou un n-uplet de valeurs, mais ne modifient pas de variable du contexte ; elles ne provoquent pas d'effet de bord.

fonction nom (<liste de paramètres>) -> < liste des types du résultat >
<corps de la fonction>
ffonction

La valeur calculée par une fonction est décrite comme suit :

-> <liste des valeurs du résultat>

Si une fonction produit une seule valeur, elle peut être utilisée dans toute expression en place d'un opérande quelconque.
Si elle produit un n-uplet de valeurs, elle ne peut être utilisée que dans une instruction d'affectation :

[<liste de variables simples>] <- nom (<liste de paramètres effectifs>)

- les actions de modifications à résultat

Elles correspondent à la composition de l'action et de la fonction :
Elles calculent une valeur ou un n-uplet de valeurs, et modifient des variables du contexte

action nom (<liste de paramètres>) -> <liste des types des résultats>
<corps de l'action à résultats>

faction

Elle ne peut apparaître que dans une instruction d'affectation, afin d'éviter des effets de bord.

exemple : Si P est une action à résultat qui modifie la variable B, le résultat de l'instruction suivante n'est pas défini :

Z <- P(A) + B

suivant l'ordre d'évaluation des termes de l'expression, le résultat peut être différent.

Certaines procédures sont prédéfinies ; elles constituent les primitives de la notation MEFIA et sont directement utilisées dans la description des algorithmes.

3.4.0.3 - Instructions

a - Affectation : le symbole d'affectation est une flèche vers la gauche : <-

. affectation de variable simple :

nom <- <expression>

. affectation de n-uplet :

[<liste de variables>] <- [<liste expressions >]

ou [<liste de variables simples>] <- nom (<liste de paramètres effectifs>).

b - Partition :

L'instruction choix permet d'établir une relation entre un ensemble de cas et un ensemble d'actions. Les cas envisagés doivent s'exclure mutuellement et correspondre à tous les cas possibles.

choix
<expression logique 1> : <groupe d'instruction 1>
...
<expression logique n> : <groupe d'instructions n >
fchoix

L'exécution d'une telle instruction consiste à exécuter le groupe d'instruction associé à l'expression logique qui prend la valeur vrai. Il y a donc erreur si aucune des expressions logiques ne prend cette valeur.

c - Itération

La forme principale d'itération est la suivante :

itérer
  <groupe d'instructions 1>
arrêt <expression logique>
  <groupe d'instructions 2>
fitérer

De cette forme on déduit les deux schémas d'itération particuliers classiques :

. tantque : <groupe d'instructions 1> est vide

tantque non <expression logique> faire
      <groupe d'instructions 2>
ftantque

. répéter : <groupe d'instructions 2> est vide

répéter
  <groupe d'instructions 1>
jusqu'à <expression logique>

3.4.0.4 - Univers

La notion d'univers est liée au souci d'exprimer les relations entre informations et actions nommées.

Remarque : La notion d'univers peut être rapprochée de celle de "module" [Par 72] et de "machine abstraite" [Gal 77].

Un univers est formé par :

a - La description algorithmique de l'univers : elle regroupe les procédures et les informations nommées interdépendantes.

b - Des "accès à l'univers" : c'est l'ensemble des noms d'actions nommées qui permettent de se référer aux informations regroupées dans l'univers. Chacun de ces noms est un "accès à l'univers" ; il est précédé du mot clé "accès" dans la description de l'univers.

univers nom
    <définition des informations nommées de l'univers>
    [accès] <définition de la procédure 1>
    ...
    [accès] <définition de la procédure n>
funivers

3.4.0.5 - Mise en page

. Un commentaire est encadré par des accolades :

{ commentaire }

. A chacune des constructions proposées ci-dessus correspond une mise en page précise permettant de mettre en évidence la structure de l'algorithme.

. Les mots clés de la notation sont soulignés.

. Le ; est le délimiteur d'instruction indiquant une exécution en séquence. Il est systématiquement omis lorsqu'il se trouve en fin de ligne ou avant un mot clé.

. A chaque schéma de traduction proposé est associé un lexique. Celui-ci définit les objets qui apparaissent dans le schéma.

3.4.1 - Le didacticiel

Le didacticiel est décrit comme l'algorithme principal ; il gère les appels aux dialogues et aux unités de liaison. De plus, il fait l'interface avec l'utilisateur, c'est à dire qu'il gère l'entrée et la sortie du didacticiel en utilisant éventuellement des mécanismes de protection (mot de passe) si nécessaire.

Sa description dans la notation MEFIA est la suivante :

```
protocole_de_début_de_session
  { opérations permettant à l'apprenant d'entrer dans le
    didacticiel : présentation de scène et dialogue éventuel avec
    l'apprenant, initialisation (chargement) des environnements
    élève et didacticiel. }
  { noeud_suivant = information de l'environnement didacticiel :
    contient le noeud du graphe à exécuter }
répéter
  noeud_courant <- noeud_suivant
  choix
    noeud_courant = 1 : D1 ; UL1
    noeud_courant = 2 : D2 ; UL2
    ...
    noeud_courant = n : Dn ; ULn
  fchoix
jusqu'à noeud_suivant = FIN ou fin_de_session

protocole_fin_de_session

  { sauvegarde de l'environnement didacticiel et présentation de
    la scène indiquant la fin de session. }
```

Lexique :

Di : représente le nom de la procédure par laquelle on a accès au dialogue Di (algorithme principal du dialogue)

ULi : représente le nom de la procédure correspondant à l'unité de liaison ULi (calcule la valeur du noeud suivant)

1,2,...,n : représentent les valeurs codées des noms des noeuds du graphe.

fin_de_session : logique dont la valeur est vrai si la fin de session est demandée par l'apprenant ou le système.

noeud_courant : variable du programme principal contenant le nom (codé) du dialogue en cours d'exécution.

noeud_suivant : variable de l'environnement didacticiel qui contient le nom (codé) du dialogue suivant à exécuter. Cette variable est modifiée par les unités de liaison.

Ce schéma est toujours le même, quel que soit le graphe à décrire. Chaque noeud est codé par un entier et décrit par les noms du dialogue et de l'unité de liaison associée. Les liens du graphe sont traduits par les valeurs que peut prendre la variable noeud courant à la fin de l'exécution de chaque unité de liaison.

Modifiabilité :

Au niveau du didacticiel, la modification la plus fréquente consiste en l'ajout ou la suppression
- d'un noeud du graphe
- d'un lien dans le graphe.

a) L'ajout ou la suppression d'un lien du graphe consiste à modifier le contenu de l'unité de liaison du noeud origine du lien : on modifie le calcul du dialogue suivant à exécuter. Cette modification n'a pas d'incidence sur la description du didacticiel (graphe).

b) L'ajout d'un noeud dans le graphe implique un cas supplémentaire dans le choix du noeud courant, et la description de la composition de ce noeud : le nom du dialogue et de l'unité de liaison qui lui est associée. La suppression d'un noeud implique la suppression du cas correspondant dans le choix du noeud courant.
Les deux cas entraînent également la modification des unités de liaison des noeuds origines d'un lien vers le noeud concerné.

Le respect de ces règles assure la correction des modifications éventuelles à apporter au graphe.

3.4.2 - L'environnement apprenant

Les informations de l'environnement apprenant sont regroupées dans un fichier (fichier apprenant) créé au niveau du système EAO. Elles sont lues au début de chaque session et regroupées dans un univers, ce qui permet d'assurer leur protection.

Le didacticiel ne fait pas de mise à jour de ces informations. Celle-ci se fera au niveau du système, par l'enseignant utilisateur.

Pour définir les informations de l'environnement apprenant, on utilise le mécanisme de définition d'information nommée de MEFIA. Cela permet de définir pour chaque information :

- son nom
- son type
- son accessibilité, c'est à dire l'autorisation d'accès à sa valeur, dans le didacticiel.

Pour chaque didacticiel à construire, l'auteur doit décrire cet environnement. Néanmoins, le système peut lui fournir un univers environnement apprenant par défaut qu'il pourra utiliser directement.

Exemple de description :

L'environnement apprenant est formé :

- du nom de l'apprenant sur 20 caractères
- de sa date de naissance formée de 3 entiers

```
univers environnement_apprenant
  { définition des informations de l'univers }
  accès tab (1:20) caractère NOM
        entier jour_naissance
        entier mois_naissance
  accès entier année-naissance
  { seuls le nom et l'année de naissance sont accessibles
  directement dans le didacticiel ; les autres informations sont
  protégées }
  accès action init_envir_apprt
    { lecture des informations de l'environnement }
        ouvrir (fichier_apprt, lecture)
        lire (fichier_apprt, NOM, jour_naissance, mois_naissance,
        année_naissance)
        fermer (fichier_apprt)
        faction
funivers
```

Les informations de l'environnement apprenant ne sont en principe pas susceptibles d'être modifiées. Dans ce cas, il faut :

- modifier la définition des informations concernées
- modifier la procédure de lecture des informations de l'environnement.

Une telle modification implique également la modification du fichier apprenant de tous les utilisateurs du didacticiel. De plus il faut prévoir les mêmes modifications de la description de l'univers pour tous les didacticiels qui partagent ce même fichier apprenant ; ceci est particulièrement important en cas de modification de l'environnement fourni par défaut.

3.4.3 - L'environnement didacticiel

Nous avons distingué deux environnements didacticiels (cf. §1.1.3) :

- l'environnement didacticiel classe
- l'environnement didacticiel apprenant

a) L'environnement didacticiel classe est mis à jour au niveau du système par l'enseignant utilisateur à partir des environnements didacticiels de chaque apprenant. Il est consultable dans le didacticiel et rangé dans un fichier lu en début de session.
Ces informations, comme pour l'environnement apprenant, ne sont pas modifiées en cours de session. La description en MEFIA est identique à celle de l'environnement apprenant. Elle se fait par un univers regroupant les informations de l'environnement et la procédure qui les initialise.

b) L'environnement didacticiel apprenant regroupe les informations spécifiées par l'auteur pour évaluer l'activité de l'apprenant dans le didacticiel. Les procédures qui manipulent ces informations sont:

- la procédure d'initialisation qui fait la lecture du fichier et initialise les informations de l'univers en début de session

- les unités de liaison qui modifient ces valeurs à la fin de chaque dialogue

- la procédure exécutée en fin de session qui sauvegarde les informations sur fichier.

La description en MEFIA se fait par un univers :

```
univers environnement_didacticiel_apprenant
  déclaration des informations de l'univers
  accès action init_e_d_a
        { lecture des informations de l'environnement didacticiel
          apprenant }
        faction

  accès action UL1
        unité de liaison
        faction

        ...

  accès action ULn
          unité de liaison
        faction

  accès action fin_env_didact
          { assure la sauvegarde des informations de l'univers ;
            procédure appelée en fin de session}
        faction

funivers
```

La modification de l'environnement apprenant entraîne la modification des descriptions des informations concernées (ajout, suppression, modification). Elle entraîne la modification des procédures d'initialisation et de recopie de l'environnement. Enfin, les unités de liaison qui manipulent les informations concernées doivent être modifiées en conséquence.

3.4.4 - Les unités de liaison

L'unité de liaison met à jour l'environnement didacticiel apprenant à partir des informations conservées au cours de l'exécution du dialogue auquel elle est associée. Elle décrit,de plus, les conditions de passage au dialogue suivant.

L'unité de liaison se décrit en MEFIA par une procédure qui modifie l'environnement didacticiel apprenant. C'est un algorithme parfois complexe (cf. exemple §1.2.4). La procédure fait partie de l'univers "environnement didacticiel apprenant" (cf. §3.4.3). Elle est accessible de l'extérieur de l'univers car elle est appelée par le programme principal.

accès action ULi

déclaration des variables locales

mise à jour de l'environnement didacticiel

calcul du prochain dialogue (modifie la variable noeud_suivant référencée dans le programme principal (cf. §3.4.1))

faction

Une modification de l'unité de liaison ne peut concerner que l'algorithme de mise à jour de l'environnement didacticiel ou de calcul du noeud suivant à exécuter.

3.4.5 - Le dialogue

Un dialogue regroupe
- des informations : l'environnement dialogue,
- des situations pédagogiques qui modifient ces informations et sont structurées en graphe.

Il gère les appels aux situations pédagogiques. En fin d'exécution, le dialogue fournit en résultat l'environnement dialogue qui est exploité par l'unité de liaison qui lui est associée. En cas d'arrêt au cours du dialogue sur demande de l'élève ou du système (fin de session), l'environnement peut être conservé sur fichier si l'auteur le souhaite.

Chaque dialogue est décrit par un univers regroupant :

- les informations de l'environnement dialogue,
- les procédures qui manipulent (modifient) ces informations : les situations pédagogiques (cf. §3.4.6) et les unités de liaison,
- une procédure qui dirige l'exécution du dialogue : elle décrit la structure de graphe du dialogue et gère les appels aux situations pédagogiques et aux unités de décision. C'est par cette procédure que l'on accède au dialogue ; c'est la seule qui soit accessible depuis l'extérieur de l'univers.

Pour traduire le graphe, on retrouve un schéma analogue au programme principal :

```
univers DIALOGUE_i
  définition des variables de l'environnement dialogue

  accès action Di
        { procédure principale : dirige l'exécution du dialogue }
        { initialisation de l'environnement, éventuellement
          lecture depuis un fichier si dialogue interrompu }
        init_env_dialogue
        répéter
          sp_courante <- sp suivante
          choix
            sp_courante = 1 : SP1 ; UD1
            sp_courante = 2 : SP2 ; UD2
            ...
            sp_courante = m : SPm ; UDm
          fchoix

        jusqu'à sp_courante = FIN ou  fin_de_session
        fin_dialogue
      faction Di

    action init_env_dialogue
      { peut être défini par défaut }
    faction

    action SP1
    faction          situations pédagogiques du dialogue   non
                     accessibles depuis l'extérieur car
    ...              spécifiques au dialogue décrit.

    action SPm
    faction

    action UD1
    faction                  unités de décision associées aux
    ...                      situations pédagogiques
```

action UDm
faction

action fin_dialogue
{ protocole de fin du dialogue ; peut être défini par défaut }
faction

autres actions nommées du dialogue utilisées par les situations pédagogiques ou d'autres actions nommées

funivers DIALOGUE_i

Lexique :

SPj	: nom de la situation pédagogique j
UDj	: nom de l'unité de décision associée à la situation pédagogique j
Di	: nom de la procédure principale du dialogue ; ce nom est utilisé dans le programme principal pour appeler le dialogue.
1,2,...,m	: représentent les valeurs codées des noms des situations pédagogiques SP1, SP2,..., SPm.

Ce schéma est toujours le même, quel que soit le dialogue à décrire.

Remarque : Les dialogues produits dans le projet MOSAIQUE ne comportent que quelques situations pédagogiques se succédant séquentiellement. Le schéma de description en est donc largement simplifié.

Une modification du graphe d'un dialogue est traitée de façon analogue à la modification du graphe du didacticiel.

3.4.6 - Les situations pédagogiques

La situation pédagogique est l'unité de décomposition minimale d'un dialogue. Elle est décrite en MEFIA par une procédure de l'univers dialogue.
C'est une action de modification car elle met à jour les variables de l'environnement dialogue (cf. §3.4.5.). Sa description est une composition d'actions pédagogiques (cf. §3.4.7.). Elle se termine par la description de l'unité de décision. Les opérateurs de composition d'actions sont ceux de la notation MEFIA : itération, partition, séquencement.

```
action SPi

    définition des informations locales à la situation pédagogique

    composition d'actions pédagogiques

    description de l'unité de décision

faction
```

La modification d'une situation pédagogique consiste à modifier la composition d'actions pédagogiques qu'elle décrit, c'est à dire l'algorithme général d'une partie de dialogue.

3.4.7 - Les actions élémentaires

Les actions élémentaires permettent de décrire de façon détaillée les interactions apprenant-système. Elles correspondent au niveau de description le plus bas.

3.4.7.1 - Les actions faisant intervenir des médias

Les actions faisant intervenir des médias posent des problèmes de portabilité.

a. La présentation de scène ou de sollicitation :

Selon les matériels disponibles on aura :

- des tailles d'écran différentes
- des codes de caractères spéciaux (accentués, semi-graphiques) différents
- des possibilités vidéo différentes
- des médias spécialisés différents
- des possibilités graphiques et (ou) sonores plus ou moins évoluées
- des possibilités d'impression différentes
- etc.

b. la réception d'informations :

Elle se fait en général par l'intermédiaire d'un clavier alpha-numérique. Celui-ci est en général différent d'un matériel à l'autre :

- les touches ne sont pas toutes les mêmes ni en même nombre
- les codes à envoyer pour obtenir certains caractères sont différents

Pour résoudre ces problèmes, nous avons défini pour chaque média, des niveaux en fonction de ses possibilités techniques. Chaque niveau englobe le niveau inférieur.
La figure 4 montre à titre d'exemple une telle hiérarchie pour un écran de visualisation :

niveau	possibilités d'envoi	caractéristiques
standard	. lignes de caractères	.nombre de lignes .nombre de colonnes (caractères par ligne écran) .caractères alphabétiques accentués ou non .noir et blanc .codage interne
vidéo	.lignes de caractères .adressage curseur .inverse vidéo .double intensité .clignotement .soulignement .effacement écran complet	idem
semi-graphique	idem vidéo + .jeu de caractères semi-graphiques standard	idem
graphique	idem vidéo + .affichage d'un point .affichage d'un segment	.nombre de points - horizontalement - verticalement
graphique couleur	idem graphique + .définition de la couleur du fond de l'écran .définition de la couleur d'affichage	idem graphique + .nombre de couleurs disponibles

(figure 4)

La notation MEFIA ne comporte pas d'instructions d'entrée ou de sortie spécifiques. Pour chaque niveau d'un média donné, nous avons défini des **primitives** correspondant aux possibilités techniques supplémentaires de ce niveau par rapport au niveau inférieur.

La description d'une présentation de scène en MEFIA se réduit à une composition de primitives. L'utilisation d'une primitive MEFIA se fait de la même façon que les autres procédures : par la spécification de son nom et des paramètres effectifs s'il y a lieu.

Pour résoudre les problèmes de synchronisation et réaliser des pauses entre les présentations de messages (temporisation), nous avons introduit une primitive d'attente.

Remarque : Pour chaque média, on se donne une primitive qui définit ses caractéristiques ; par exemple, nombre de lignes et de colonnes, niveau d'un écran. Cette primitive aura pour effet d'initialiser les informations qui caractérisent l'état du média, et de permettre des contrôles à l'exécution.

Cette démarche est également valable pour les médias permettant une réception d'information : on définit une primitive de réception adaptée à chaque média. Pour un clavier, on définit en général une primitive de réception d'information de nature alpha-numérique dont l'écho se fait de façon contrôlée dans une zone de l'écran.

3.4.7.2 - La mise sous forme canonique et l'analyse de réponse

Ces opérations manipulent essentiellement :

- les réponses de l'apprenant,
- les réponses attendues par l'auteur.

Une action de mise sous forme canonique est un calcul effectué sur une information, en vue d'en obtenir une forme différente (normalisée), et de plus des informations complémentaires utiles à l'analyse de réponse (comme par exemple le nombre de mots utilisés ou la longueur du message). Le calcul est effectué à partir d'informations propres à l'opération de mise sous forme canonique (par exemple un alphabet).

La description en MEFIA se fait par un univers :

- les objets de l'univers sont les informations propres à la mise sous forme canonique,

- les procédures qui manipulent ces objets effectuent la mise sous forme canonique.

Une seule procédure est accessible depuis l'extérieur de l'univers: la procédure principale décrite sous la forme d'une action de modification à résultats. C'est elle qui porte le nom de l'action de mise sous forme canonique et qui produit les résultats du calcul.

univers MISE_SOUS_FC

définition des informations locales à la mise sous forme canonique

accès action NOM_FC (type INF) -> liste types

objets locaux à l'action

algorithme principal de calcul de la forme canonique

faction

procédures annexes nécessaires à la spécification de l'algorithme principal

funivers

Lexique :

NOM_FC : nom de l'action de mise sous forme canonique
INF : nom formel de l'information à transformer
type : type de l'information à transformer
liste types : liste des types des résultats produits.

Utilisation :

[R1, R2,..., Rn] <- NOM_FC (REPONSE)

REPONSE : information à transformer
R1, R2,...,Rn : noms des variables récupérant les résultats de la mise sous forme canonique.

Remarque : l'univers décrit ici est indépendant du reste du didacticiel.

Une analyse de réponse est un calcul effectué sur une information éventuellement préparée par une mise sous forme canonique. Ce calcul détermine la coincidence entre cette information et le modèle donné par l'auteur. Le résultat produit est un ensemble de valeurs qui décrivent l'état de la réponse (exemple : nombre de fautes d'orthographe, nombre de mots omis, etc..). Cet ensemble est appelé vecteur d'état [ADI 81].

La description de l'analyse de réponse se fait par un univers structuré de façon similaire à une mise sous forme canonique.

Intégration à la notation MEFIA

Etant donnée la multiplicité des mises sous forme canonique et des analyses de réponse possibles, il n'a pas été défini de primitives prédéfinies en MEFIA leur correspondant.
Ce n'est que lorsque l'une de ces opérations est reconnue comme étant classique, c'est-à-dire rencontrée dans de nombreux dialogues ou didacticiels, qu'elle est intégrée à la notation MEFIA. L'univers correspondant devient alors un univers prédéfini, et le nom de l'action principale devient une primitive de la notation.
L'intégration à MEFIA de primitives adaptées aux médias utilisés ou résolvant des problèmes spécifiques à l'EAO en fait une notation adaptée à la description des didacticiels, notamment la description des situations pédagogiques. On appellera MEFIA-EAO, la notation obtenue par l'intégration de ces primitives spécialisées.

Les problèmes de portabilité

Pour ces opérations, le problème de portabilité se pose au niveau du code interne de certains caractères non standards, comme par exemple les caractères accentués, qui changent d'une machine à l'autre. Un codage interne de tous les caractères spéciaux utilisés par le didacticiel est nécessaire. Ce codage se fait au cours de la réception d'informations. Il facilite également la portabilité des primitives d'affichage à l'écran.

3.4.7.3 - Les actions de gestion des informations

Les actions de gestion des informations sont utilisées par les situations pédagogiques pour la mise à jour de l'environnement dialogue (cf. §1.2.2). Ce sont des procédures de l'univers dialogue (cf. §3.4.5) décrites à la suite des situations pédagogiques.
La description d'une action de gestion d'informations se fait par une action de modification décrite en termes d'instructions élémentaires MEFIA.

action GESTION_INFO

définition des objets locaux à la procédure

description des opérations de gestion des informations (composition d'instructions élémentaires MEFIA).

faction

Remarque : Certaines de ces actions manipulent des fichiers pour la conservation de traces.

3.4.7.4 - L'exécution de programmes externes

Certains traitements externes, conçus et réalisés indépendamment du didacticiel (modules de simulation, consultation de bases de données, etc...), doivent pouvoir être facilement utilisés depuis un dialogue. Pour cela il est nécessaire de définir une forme standard de communication.

Forme standard utilisée :

Tout programme externe doit être appelé de la même façon. On connaît en général les informations suivantes :
- le nom du programme externe
- le nombre, le type et l'ordre des paramètres à transmettre
- le nombre, le type et l'ordre des résultats produits.

Description proposée en MEFIA pour assurer la communication :

Une primitive MEFIA-EAO "TRAITEMENT-EXTERIEUR" assure la communication. Elle fait l'interface entre le didacticiel et l'extérieur.

- Informations fournies à la primitive :

  . le nom du programme externe,
  . le nombre de paramètres,
  . la liste des types (codés) des paramètres,
  . les valeurs des paramètres effectifs,
  . le nombre des résultats attendus,
  . la liste des types (codés) des résultats.

- Informations produites par la primitive :

La primitive fournit les résultats produits par le programme externe.

Cette primitive assure l'indépendance par rapport au programme externe utilisé.

Remarque : L'application Français langue Etrangère du projet MOSAIQUE n'a pas donné lieu à l'utilisation de programmes externes.

3.4.8 - Les requêtes

Nous avons vu que dans chaque dialogue, on met à la disposition de l'apprenant un certain nombre de requêtes. Des informations pédagogiques qui permettent de les gérer sont contenues dans l'environnement dialogue :

- des variables logiques signalant si une requête est inhibée ou non,
- des variables dont la durée de vie est supérieure à celle de l'appel d'une requête (exemple : nombre d'appels de chaque requête),

Ces informations sont modifiées par les actions d'inhibition ou d'activation d'une requête, et par la procédure qui appelle la requête demandée. Ces actions font donc partie de l'univers de dialogue :

```
accès action REPONSE_REQUETE (reponse REP)

        définition des objets locaux
        discrimination entre requête et réponse
        appel de requête s'il y a lieu

    faction


accès action INHIBER (requête REQ)

        inhibition de la requête REQ

    faction


accès action ACTIVER (requête REQ)

        activation de la requête REQ

    faction
```

Nous avons distingué trois types de requêtes (cf §2.2.7).

Une requête globale, indépendante d'un dialogue, est décrite par un univers de même structure qu'une analyse de réponse : seule la procédure principale portant le nom de la requête est accessible de l'extérieur.

Une requête spécifique à un dialogue (requête globale particulière ou requête locale) est intégrée à l'univers décrivant le dialogue et se décrit comme une situation pédagogique.

3.4.9 - Conclusion

Les règles que nous venons de présenter permettent une structuration informatique très modulaire d'un didacticiel. Ceci favorise la modification aisée d'un objet, sans conséquences pour les autres modules.
Ces règles ont permis de mettre en évidence les relations entre les modules, et l'indépendance de certains par rapport à d'autres. Enfin, elles ont permis de définir la notation MEFIA-EAO par l'intégration de primitives adaptées aux médias utilisés et à des opérations spécifiques aux didacticiels, en tenant compte des problèmes de portabilité.

Ces règles ont été expérimentées dans le cadre du projet MOSAIQUE, ce qui a permis l'obtention d'une forme unique de description du didacticiel, lisible et facilement modifiable. Cette expérimentation a également permis de définir un certain nombre de schémas d'analyse de réponse et de mise sous forme canonique, utilisés fréquemment dans la description des dialogues. Ces schémas ont été intégrés à la notation MEFIA-EAO. On a ainsi obtenu un ensemble complet de primitives, correspondant à toutes les actions pédagogiques apparaissant couramment dans la description des dialogues.

D'autre part, les principales modifications à effectuer, concernent les présentations de scènes, notamment le contenu ou la position à l'écran des messages informatifs. Afin d'améliorer la modifiabilité du didacticiel, nous avons séparé du programme ces informations. En conséquence, il y a chargement de ces informations en mémoire, au début de l'exécution de chaque dialogue. Leur manipulation par les actions pédagogiques élémentaires se fait au travers de références (noms) qui leur sont associées. Tout dialogue produit est composé d'une partie "programme" (actions pédagogiques) et d'une partie "données" (informations pédagogiques).

3.5 - Traduction systématique vers un langage de programmation disponible sur micro-ordinateur

3.5.1 - La programmation sur micro-systèmes

La grande majorité des constructeurs de micro-systèmes offre essentiellement les possibilités d'utilisation suivantes de leurs matériels :

- la programmation dans le langage d'assemblage du micro-processeur autour duquel est construit le micro-ordinateur,

- la programmation en langage BASIC (interprété ou compilé),

- la programmation en langage évolué, essentiellement PASCAL.

3.5.1.1 - La programmation en langage d'assemblage

La programmation en langage d'assemblage présente l'avantage d'offrir au programmeur l'accès à des ressources non disponibles en langage plus évolué, comme par exemple la gestion des interruptions. Elle permet de plus une exécution rapide des programmes. Elle présente toutefois un certain nombre d'inconvénients :

- il n'y a qu'une **portabilité partielle** des programmes : on est tributaire du micro-processeur et du système d'exploitation disquettes (s.e.d) disponibles. On ne profite pas des facilités (gestion des entrées/sorties et des fichiers) offertes par le système et utilisables facilement depuis les autres langages. Il en résulte une portabilité effective uniquement entre machines de même série et de même marque, équipées du même s.e.d.

- il est **plus difficile** de programmer en langage d'assemblage de façon **fiable** : les temps de mise au point et de maintenance sont très importants. Ceci est dû au niveau de détail auquel le programmeur est contraint. Il en ressort un besoin d'avoir des outils plus puissants afin de compenser ces défauts.

3.5.1.2 - La programmation en langage BASIC

BASIC est le langage disponible sur la grande majorité des micro-systèmes [CTI 81], et encore souvent l'outil de programmation le plus évolué.

Caractéristiques du langage BASIC

BASIC est avant tout un langage conçu pour les débutants en informatique [Leb 80]. Il peut être appris très vite par des néophytes. En général, l'utilisateur dispose d'un "interpréteur BASIC" qui lui permet

- d'introduire ou de charger des programmes en machine,
- d'exécuter le programme chargé en mémoire,
- de modifier son programme en ajoutant ou supprimant des lignes,
- d'interrompre à tout moment l'exécution de son programme et de pouvoir consulter le contenu des variables du programme, ce qui facilite la mise au point.

Tout se déroule de façon interactive avec l'utilisateur : entrée des programmes, détection des erreurs, exécution, consultation des variables.

Un programme BASIC est une suite de lignes d'instructions numérotées en ordre croissant. C'est un programme totalement ouvert, c'est à dire qu'il n'y a pas de notion de bloc. Toutes les variables sont accessibles dans tout le programme.

On ne peut manipuler que deux ou trois types différents de variables : des réels, des chaînes de caractères et éventuellement des entiers.

La programmation en BASIC offre quelques intérêts :

- Par rapport à la programmation en langage d'assemblage, les algorithmes sont exprimés plus facilement.
- Son aspect interactif permet le test rapide des algorithmes.
- L'utilisation des fonctions du s.e.d disponible est simple.
- C'est un langage très utilisé, disponible sur la grande majorité des micro-systèmes.

En BASIC compilé, on perd l'aspect interactif, mais les programmes produits sont plus concis et beaucoup plus performants. De plus, la place mémoire disponible à l'utilisateur apprenant est augmentée de la taille de l'interpréteur.

BASIC n'en reste pas moins un langage pauvre :

- Le nom des objets BASIC est limité à deux caractères ; de ce fait on a :
  - un nombre limité de variables possibles
  - un manque de clarté des programmes (mauvaise lisibilité des programmes)

- Le concept d'action nommée [SCM 80], rencontré dans les langages plus évolués est difficilement exprimable du fait de l'absence de la notion de bloc. Il en résulte un manque de sécurité des programmes.

. BASIC n'étant pas un langage déclaratif, il n'y a pas de types nommés. De ce fait, la spécification des informations est plus difficile à exprimer.

. Les seuls types d'objets manipulables sont en général les réels et les chaînes de caractères ; les structures de données définissables sont les tableaux à une, deux ou trois dimensions. Souvent, la deuxième et la troisième dimension sont limitées, et les intervalles d'adressage ont toujours pour borne inférieure la valeur zéro.

3.5.1.3 - La programmation en langage PASCAL

La programmation en langage PASCAL [Wir 71.a] est beaucoup plus claire et plus fiable que les modes de programmation présentés ci-dessus. Lorsque le langage est compilé, elle permet d'obtenir des programmes performants. Ceci n'est plus vrai lorsque l'on n'est pas en présence d'un véritable compilateur. C'est le cas avec de nombreux micro-ordinateurs : le compilateur PASCAL produit un code interprété à l'exécution, d'où une baisse sensible des performances.

PASCAL présente toutefois deux inconvénients pour la traduction des didacticiels :

- il n'y a pas, de façon standard, des instructions d'entrée/sortie interactives correctes : on est obligé en général de se créer des actions d'entrée/sortie écrites dans un autre langage, le plus souvent en langage machine.

- il n'y a pas de type chaîne (string) dans la version standard du langage, ce qui augmente la difficulté de description des opérations qui manipulent des informations alpha-numériques.

- la notion de module n'existe pas de façon standard.

3.5.2 - Le choix du langage de programmation

Les programmes à produire doivent s'exécuter sur micro-ordinateur. Nous devons choisir entre PASCAL, BASIC et le langage d'assemblage du micro-processeur.

Des trois langages, PASCAL nous paraît le mieux adapté du point de vue de la lisibilité, de la fiabilité et de l'efficacité des programmes produits.
Au niveau du projet MOSAIQUE ce choix a pourtant été écarté pour les raisons suivantes :

- Les programmes produits doivent être facilement diffusables. Au départ du projet, un tiers seulement des micro-ordinateurs sur le marché français offrent la possibilité de travailler en PASCAL, alors que tous disposent du BASIC [CTI 81].

- Le micro-ordinateur sur lequel doit se faire l'expérimentation du didacticiel produit (Goupil 2), dispose d'une version PASCAL très incomplète qui ne permet pas notamment la compilation séparée. D'autre part, la version UCSD de PASCAL produit un code interprété et demande une importante mémoire de masse.

- Nous voulons expérimenter les méthodes de traduction systématique de programmes de taille importante, décrits dans la notation MEFIA, vers un langage pauvre.

Le choix du langage de programmation pour la réalisation du didacticiel se porte donc sur BASIC.

Des problèmes de portabilité restent à résoudre : en effet, bien que tous les constructeurs de micro-systèmes proposent BASIC, et que la plupart des interpréteurs et des s.e.d. disponibles soient construits par la même société américaine (Microsoft), des problèmes de portabilité se posent. On trouve dans [Adm 79] une étude de ces différences qui sont de quatre ordres :

- différences de puissance d'expression
- différences de performances
- différences de syntaxe de certaines instructions
- absence ou présence de certaines fonctions ou instructions spécialisées.

Cette étude a permis de dégager un noyau du langage présent sur toutes les versions rencontrées. L'utilisation de ce noyau permet la production de programmes portables.

3.5.3 - La traduction systématique de la notation MEFIA-EAO

Nous avons défini des règles de traduction systématique de la notation MEFIA vers le langage BASIC. Seul le noyau portable mis en évidence, est utilisable pour traduire les objets décrits en MEFIA. Les seules instructions BASIC qui composent ce noyau sont :

LET
IF
GOTO
GOSUB
REM
RETURN
STOP
END
les fonctions : CHR$, STR$, LEN, VAL, plus quelques fonctions mathématiques
et l'instruction de définition de tableaux : DIM

Les autres instructions ou fonctions rencontrées présentent des différences syntaxiques ou sémantiques d'une version de BASIC à une autre.

Exemple : Différence sémantique de l'instruction d'itération FOR NEXT :

syntaxe : FOR v = début TO, fin STEP pas
composition d'instructions
NEXT v

Suivant la version de BASIC rencontrée le schéma algorithmique correspondant sera :

```
v <- début                          ou      v <- début
tantque v <= fin faire                      répéter
  composition                                 composition
  d'instructions                              d'instructions
  v <- v + pas                                v <- v + pas
ftantque                                    jusqu'à v > fin
```

Principes de la traduction vers BASIC

- Tout schéma itératif se traduit avec les instructions IF et GOTO

- Toute procédure nommée se traduit par une subroutine, le contenu de la subroutine étant une composition d'instructions du noyau. Pour chacune, il faut définir les noms des variables BASIC qui représentent les paramètres.

- Le passage des paramètres du programme appelant à la subroutine se fait par valeur : avant l'appel de la subroutine, les valeurs des paramètres effectifs sont copiées dans les variables qui correspondent aux paramètres. On procède de façon symétrique pour les résultats produits : un résultat est rangé dans une variable BASIC ; sa valeur est récupérée dans le programme appelant immédiatement après l'appel.

. Le programme produit étant entièrement ouvert, il faut gérer très proprement les noms des variables BASIC utilisées. Il faut notamment distinguer les variables du didacticiel de celles utilisées pour traduire les primitives MEFIA-EAO, ceci afin d'éviter les effets de bord.

. Les primitives de la notation MEFIA-EAO sont traduites comme les autres procédures nommées, par des subroutines. Leur contenu est en général exprimé dans le noyau BASIC, en termes de primitives de plus bas niveau.

Règles générales pour assurer la portabilité

Toute instruction BASIC non standard ne doit apparaître que dans une subroutine et une seule. La réunion des subroutines contenant des instructions non standards, forme le **noyau BASIC de base** à partir duquel sont décrites toutes les autres subroutines, notamment les primitives MEFIA-EAO.
Le passage d'une version de BASIC à une autre implique simplement la réécriture de ce noyau.

. Un programme BASIC est une suite de lignes numérotées en ordre croissant. L'appel d'une subroutine se fait par l'intermédiaire d'un numéro de ligne et non par un identificateur comme pour les autres langages. En conséquence, chaque primitive MEFIA-EAO doit être codée par un numéro de ligne BASIC ; il faut réserver des numéros de lignes pour les primitives et le noyau non portable. On obtient ainsi une bibliothèque de procédures BASIC.
Le programme exécutable est obtenu par la concaténation de la partie décrite en BASIC standard à cette bibliothèque.

3.6 - Résultats de l'expérimentation

Les méthodes proposées ont été largement expérimentées pour la réalisation manuelle d'une première version du didacticiel. Cette expérience a permis l'obtention :

- d'une quinzaine d'heures de cours, ce qui correspond à la réalisation d'une quarantaine de dialogues;

- d'un ensemble de primitives MEFIA-EAO réalisées en BASIC, constitué au fur et à mesure de l'expérience acquise ; il permet une diminution importante du temps d'écriture et de mise au point d'un dialogue : un dialogue dont la réalisation prenait une semaine au début du projet, ne demande plus qu'une à deux journées de travail à l'informaticien, suivant sa complexité.

Ceci démontre l'efficacité d'une attitude systématique dans l'utilisation d'un langage de programmation, même quand celui-ci est pauvre.

Cette réalisation a également permis de mettre en évidence des problèmes dans la production :

- les limites de BASIC pour la production de logiciels conséquents,
- les problèmes d'implantation des programmes produits sur des micro-ordinateurs, dûs aux limites physiques de ces machines et à la masse des programmes produits à gérer.

3.6.1 - Les limites de BASIC

L'utilisation intensive de BASIC a permis de dégager les problèmes suivants :

- La numérotation figée des lignes de programmes nuit à la modifiabilité : une insertion d'instruction n'est pas toujours possible directement : il faut, soit renuméroter une partie du programme, soit faire un branchement inconditionnel vers les instructions placées ailleurs dans le programme. Cette dernière solution nuit à la modifiabilité et à la lisibilité du programme.

. La limitation des noms de variables à deux caractères (le second étant obligatoirement un chiffre), impose d'avoir parfois à utiliser le même nom pour désigner des entités logiques différentes. Le choix des noms se fait en respectant des règles précises :

- Un sous-ensemble de noms d'objets BASIC est réservé à la réalisation du didacticiel, un autre à celle des primitives.

- Pour chacun des deux sous-ensembles, des noms sont réservés aux résultats, des calculs intermédiaires locaux à la procédure décrite, c'est à dire à des calculs au cours desquels aucune procédure n'est appelée. Ces objets peuvent être réutilisés sans danger dans toute autre procédure, pour conserver le même type de résultats.

- Un nom de variable locale à la procédure donnée, ne peut être utilisé dans la description d'une procédure appelée ni d'une procédure appelante.

De cette utilisation multiple de certains objets BASIC, il résulte une baisse de la lisibilité, de la fiabilité et des possibilités de modification de ces programmes élaborés manuellement.

. Le fait de n'avoir que des objets de type réel et chaîne oblige à utiliser les réels pour représenter les entiers et les logiques. De ce fait, il n'y a pas de vérification de compatibilité entre les objets de ces types d'où un manque de sécurité supplémentaire.

. Dans un programme BASIC interprété, **les commentaires coûtent cher en place mémoire.** Ceci limite leur utilisation et nuit à la **lisibilité**, notamment au niveau de la réalisation des procédures nommées et de la composition des algorithmes : itérations, analyses par cas, appels de procédures.

. La traduction de procédures récursives est complexe : chaque variable locale et chaque paramètre doit être traduit par une pile ; l'empilement des adresses de retour est assuré par BASIC.

L'ensemble de ces problèmes justifie l'étude et la réalisation d'outils qui facilitent la production des programmes.

3.6.2 - Les problèmes d'implantation sur micro-ordinateur

Le didacticiel produit doit pouvoir s'exécuter sur des micro-ordinateurs. La configuration minimale d'utilisation à laquelle nous nous intéressons correspond aux postes de travail livrés à l'Education Nationale. Elle comprend :

. niveau du matériel :

- une unité centrale équipée d'un microprocesseur 8 bits
- une mémoire centrale de 64K octets (mémoire vive)
- un écran de visualisation avec attributs vidéo
- un clavier alpha-numérique aux normes françaises
- deux unités de disquettes (5 pouces) d'une capacité minimale de 150K octets chacune

. au niveau logiciel :

- un système d'exploitation disquettes (s.e.d.)
- un interpréteur ou un compilateur du langage BASIC.

Le matériel utilisé pour l'expérimentation correspond à ces critères : il s'agit d'un Goupil 2 construit par la société SMT, doté d'une capacité mémoire de 64K octets, d'un microprocesseur Motorola 6808 (8 bits), et du s.e.d. FLEX qui supporte un interpréteur et un compilateur BASIC. Les disquettes utilisées sont de format 8 pouces et d'une capacité de 1 Méga octets chacune. Leur exploitation a toutefois été organisée en vue d'un transport sur des configurations à disquettes de plus faible capacité (5 pouces simple densité).

Ce type de matériel pose des problèmes d'implantation, tant du point de vue du matériel que du point de vue du logiciel disponible.

3.6.2.1 - Les problèmes de place mémoire

- Le fait d'utiliser un interpréteur BASIC limite l'utilisateur en place mémoire : l'interpréteur est en général résident, ainsi qu'une partie du s.e.d. Il ne reste que peu de place au programme utilisateur : de 20 à 40K octets suivant les logiciels disponibles.
  Le didacticiel produit doit donc être structuré en segments de programme pouvant se recouvrir en mémoire. Il faut pour réaliser cette structuration, tenir compte des possibilités de recouvrement offertes par le logiciel (cf chapitre 5).

- Chaque segment de programme BASIC exécutable (dialogue ou situation pédagogique) est obtenu par la concaténation du programme décrit en BASIC standard aux primitives MEFIA-FAO (cf §3.5.3). De ce fait, les programmes produits ont en commun toutes les primitives de la bibliothèque. Ceci présente l'inconvénient d'avoir dans chaque programme produit, des primitives inutilisées, donc un gaspillage de la mémoire utilisateur.
  La réalisation de certains programmes de taille importante nécessite la création manuelle d'une sous-bibliothèque ne comprenant que des primitives effectivement utilisées. Ceci pose le problème du calcul de la liste complète des primitives nécessaires à l'exécution du programme et diminue la fiabilité. Un outil qui réalise le calcul s'avère nécessaire.

L'interpréteur BASIC disponible pour l'expérimentation est nettement insuffisant pour supporter le didacticiel produit :

- La mémoire utilisateur n'est que de 27K octets, ce qui impose de faire du recouvrement au niveau des situations pédagogiques d'un dialogue.

- Le recouvrement disponible est très rudimentaire : il consiste en un chargement en mémoire de programmes indépendants ; il y a destruction des variables élaborées par le programme précédent. La transmission d'informations d'un programme à l'autre doit se faire au moyen d'un fichier de données, ce qui augmente très sensiblement le temps de chargement.

L'utilisation de l'interpréteur est abandonnée au profit du compilateur BASIC qui présente les intérêts suivants :

- la mémoire utilisateur est de 44K octets ;
- les commentaires ne coûtent pas de place mémoire ;
- le recouvrement des programmes se fait sans destruction des variables, ce qui permet un recouvrement au niveau soit des dialogues, soit des situations pédagogiques, et accélère le chargement ;
- le temps d'exécution d'un programme est beaucoup plus court que sous l'interpréteur : pour les calculs, le gain obtenu est environ d'un facteur 40.

Le code objet produit n'est pas translatable ; le programmeur doit gérer la mémoire en choisissant les adresses d'implantation des données et des programmes. Il y a possibilité de compiler des programmes séparément, mais leur appel se fait par une instruction BASIC non standard, en spécifiant l'adresse en mémoire du point d'entrée.

3.6.2.2 - Les problèmes de gestion de la masse des programmes produits

L'ensemble des programmes produits au cours de l'expérimentation est formé :

- du didacticiel composé d'une quarantaine de dialogues,
- d'un ensemble de primitives spécialisées, commun à tous les dialogues : la bibliothèque MEFIA-EAO.

a) **La gestion du didacticiel produit**

Chaque dialogue est composé :

- d'une partie programme composée d'un ou plusieurs fichiers suivant la taille du dialogue,
- d'un ensemble d'informations manipulées par le programme, contenues dans un ou plusieurs fichiers suivant la taille du dialogue, en général, on a un fichier d'informations par fichier programme,
- d'un ensemble de messages sonores présentés à l'apprenant au cours du dialogue,
- d'un ensemble de diapositives présentées à l'apprenant au cours du dialogue.

Ces objets sont rangés sur des supports aux possibilités de stockage limitées :

. 150K octets par disquette,
. 50 messages sonores par cassette,
. 80 diapositives par panier.

Le didacticiel se trouve donc composé physiquement d'un ensemble de disquettes, de cassettes et de paniers de diapositives. Il faut se donner les moyens de gérer ces volumes, c'est à dire être capable de spécifier à l'apprenant les changements de disquette, de cassette ou de panier de diapositives à effectuer avant l'exécution d'un dialogue donné.

Cette fonction est prise en charge par le programme principal : le didacticiel. On tient compte de cette gestion dans la traduction en BASIC du lancement de l'exécution d'un dialogue. Un ensemble de primitives facilitant cette gestion de support est défini. Ces primitives et les informations qu'elles manipulent constituent le **système de gestion de fichiers** (S.G.F.) du didacticiel produit.

b) La gestion de la bibliothèque de primitives

La bibliothèque des primitives est concaténée à la partie de dialogue décrite en BASIC standard, pour former un programme complet interprétable ou compilable.
Des modifications dans cette bibliothèque posent des problèmes et coûtent cher en opérations de mise à jour :

- Toute modification d'une primitive entraîne, en général, la mise à jour de tous les programmes du didacticiel :

  - chaque programme doit être recréé par concaténation de la nouvelle version de la bibliothèque au programme ; cette opération n'est pas nécessaire s'il y a possibilité, à l'exécution, de charger séparément des morceaux de programme en mémoire. Dans ce cas, aucune modification n'est à faire dans les programmes. Cette possibilité est néanmoins assez rare (cf. chapitre 5) dans les versions de BASIC rencontrées couramment,

  - en cas d'utilisation d'un compilateur, tous les programmes doivent être recompilés.

- Le changement du numéro de ligne BASIC correspondant au nom (point d'entrée) d'une primitive donnée n'est pas possible : il entraînerait la modification de chaque appel de la primitive dans tous les programmes du didacticiel et dans toutes les primitives.

Des outils qui facilitent cette gestion s'avèrent indispensables.

3.6.3 - Conclusion

L'ensemble des problèmes que nous venons de présenter, justifient l'étude et la réalisation d'un ensemble d'outils

- **qui améliorent la production** en libérant l'informaticien des limites du langage BASIC en lui fournissant des moyens d'expression plus puissants, et en rendant plus souple l'utilisation des primitives et plus facile leur mise à jour ;

- **qui facilitent la gestion des volumes** supportant le didacticiel, en cours d'exécution ;

- **qui permettent une meilleure utilisation de la mémoire** par l'automatisation de la construction de sous-ensembles de primitives. Une stratégie de recouvrement des programmes doit être définie en fonction des possibilités offertes par le logiciel disponible, et éventuellement des outils qui facilitent la mise en place de cette stratégie.

CHAPITRE IV

SPECIFICATION ET REALISATION D'UN ENVIRONNEMENT DE PRODUCTION DE DIDACTICIELS

Les problèmes mis en évidence dans le chapitre précédent, indiquent le besoin de se doter d'un ensemble d'outils pour améliorer la production des didacticiels. Nous présentons dans cette partie les outils spécifiés et réalisés dans le cadre du projet MOSAIQUE.
Dans le chapitre suivant, nous présentons les méthodes et les outils mis en oeuvre pour résoudre les problèmes d'implantation des didacticiels produits sur des micro-ordinateurs.

Dans le cadre du projet MEFIA [SCG 78], un environnement d'aide à la production de programmes a été défini [Dub 78] [Cas 79]. L'adaptation de cet environnement au contexte particulier des micro-ordinateurs a également été étudié [Adm 80], afin de répondre aux problèmes de programmation de ce type de matériels (cf §3.5.1).

La spécification de l'environnement proposé ici, s'appuie sur cette étude. Le noyau de l'environnement est un traducteur de la notation MEFIA-EAO vers le langage BASIC.

Après avoir présenté la structure générale et les fonctionnalités de l'environnement proposé, nous présentons chacun de ses composants.

4.1 - Stucture générale de l'environnement de production

L'environnement de production proposé est basé sur l'utilisation de la notation MEFIA-EAO pour la description des objets. Il est composé des outils suivants :

- Un analyseur de programmes MEFIA :
  il effectue l'analyse lexicographique et syntaxique du programme, vérifie les contraintes contextuelles et produit une forme interne du programme.

- Un documenteur de programmes :
  il permet d'obtenir des documents sur un programme MEFIA-EAO analysé :

. la liste complète du programme, présentée en fonction de la syntaxe

. la table des références croisées (index)

. le dossier de programmation

Ces documents sont produits à partir de la forme interne du programme.

- Un traducteur vers le langage BASIC :

il effectue, à partir de la forme interne du programme, la traduction en langage BASIC standard. Le BASIC produit n'est pas exécutable : il comporte

. des références externes non résolues : les références aux primitives utilisées,

. des directives destinées à l'éditeur de liens pour résoudre les références en avant.

- Une bibliothèque de primitives MEFIA-EAO prédéfinies :

la bibliothèque est constituée des primitives réalisées en langage BASIC, pour faciliter la description des didacticiels. Le noyau de cette bibliothèque constitue la partie non standard des programmes produits. Il est constitué de primitives décrites avec des instructions spécifiques à la version de BASIC et à la machine utilisée. Il y a autant de versions du noyau que de machines cibles.

Un ensemble d'outils est nécessaire pour gérer cette bibliothèque et assurer sa cohérence par rapport au reste du système.

- Un éditeur de liens :

L'éditeur de liens transforme le programme BASIC produit en un programme exécutable : il concatène au programme produit les primitives de la bibliothèque utilisées, et résoud les références externes (références aux primitives), et également les références en avant. Il utilise pour cela la version de la bibliothèque correspondant à la machine cible. Le programme ainsi transformé est compilable ou interprétable sur la machine cible. C'est donc l'édition de liens qui assure la portabilité des didacticiels produits.

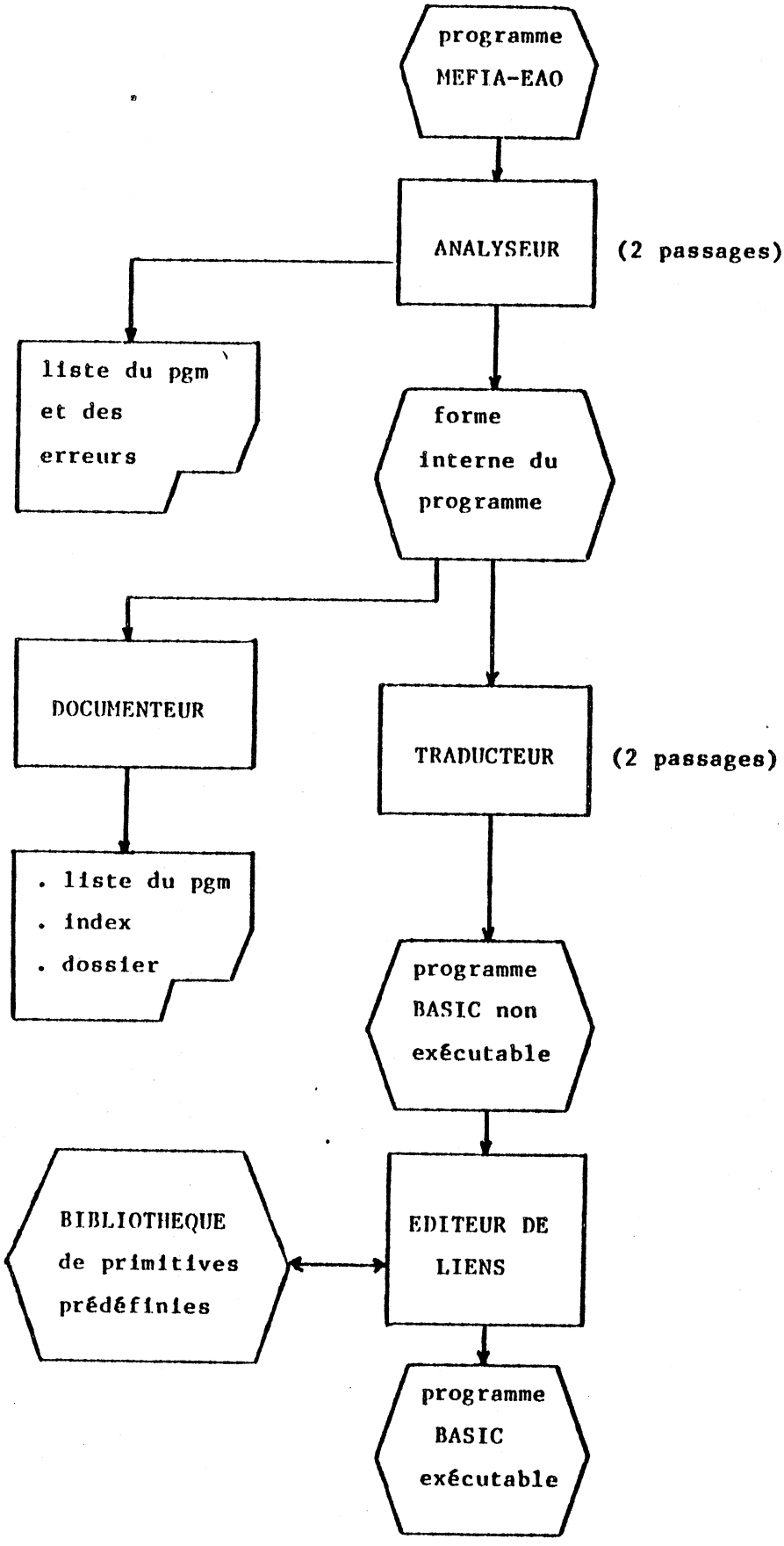
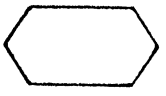


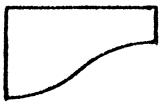
figure 5 :
Schéma général de production à partir de l'environnement défini

Conventions :

programme de traitement d'informations



informations traitées



listes résultats

. vers un programme : données en entrée

. vers une information : information produite ou modifiée

Utilisation du système de production (figure 5) :

A partir de la description du programme dans la notation MEFIA-EAO, l'analyseur produit la forme interne ainsi qu'une liste du programme et des erreurs détectées, l'utilisateur peut ensuite :

- produire des documents relatifs à son programme à l'aide du **documenteur**,
- produire la traduction de son programme en BASIC standard par le **traducteur**.

Par l'**éditeur de liens**, il rend le programme objet exécutable sur la machine cible qu'il spécifie.

La forme interne est un élément essentiel de l'environnement de production, elle permet une grande souplesse, notamment au niveau de la définition de nouveaux composants de l'environnement : pour effectuer la traduction vers un autre langage de programmation, il suffit de réaliser un nouveau traducteur et d'écrire les primitives dans ce nouveau langage. Cette technique d'utilisation d'une forme interne pour la définition de plusieurs composants d'un système devient classique en Génie Logiciel [MRR 82].

4.2 - L'analyseur de programmes MEFIA-EAO

4.2.1 - Structure générale

L'analyse est réalisée en deux passages (Figure 6) :

- premier passage : analyse syntaxique et production d'une forme interne du programme

  données : programme source
  résultats : - forme interne du programme
  - liste du programme source
  - liste des erreurs lexicographiques et syntaxiques rencontrées
  - liste des objets non définis dans le programme

- deuxième passage : analyse contextuelle du programme

  données : forme interne du programme
  résultats : liste des erreurs de nature contextuelle détectées

Le deuxième passage assure la cohérence du programme, c'est-à-dire le respect des règles de syntaxe contextuelles de MEFIA. Ces vérifications permettent la détection statique des erreurs, et de certains effets de bord pouvant provoquer des erreurs à l'exécution.

Remarque : l'analyse contextuelle du programme implique des vérifications syntaxiques non exprimables par une grammaire hors contexte. Ce type d'analyse est parfois appelé analyse sémantique par opposition à l'analyse syntaxique (aspect hors contexte).

L'analyse contextuelle n'est exécutée que si aucune erreur n'est détectée au premier passage ; le choix d'aller le plus loin possible dans la détection des erreurs n'est pas retenu : nous préférons donner à l'utilisateur la possibilité de corriger immédiatement ses erreurs plutôt que d'exécuter systématiquement l'analyse contextuelle.

Remarque : La production de la liste des objets non définis dans le programme peut se faire au deuxième passage : il s'agit d'erreurs contextuelles. Le choix de faire ce calcul à la fin du premier passage est dû à des raisons d'efficacité : toutes les informations nécessaires à la production de cette liste sont élaborées à la fin du premier passage. On évite ainsi l'exécution du second passage et la production d'erreurs en cascade si des objets non définis sont trouvés.

D'autre part, la forme interne contient la liste des primitives MEFIA-EAO (références externes) utilisées par le programme. L'élaboration de cette liste se fait en même temps que celle des objets non définis.

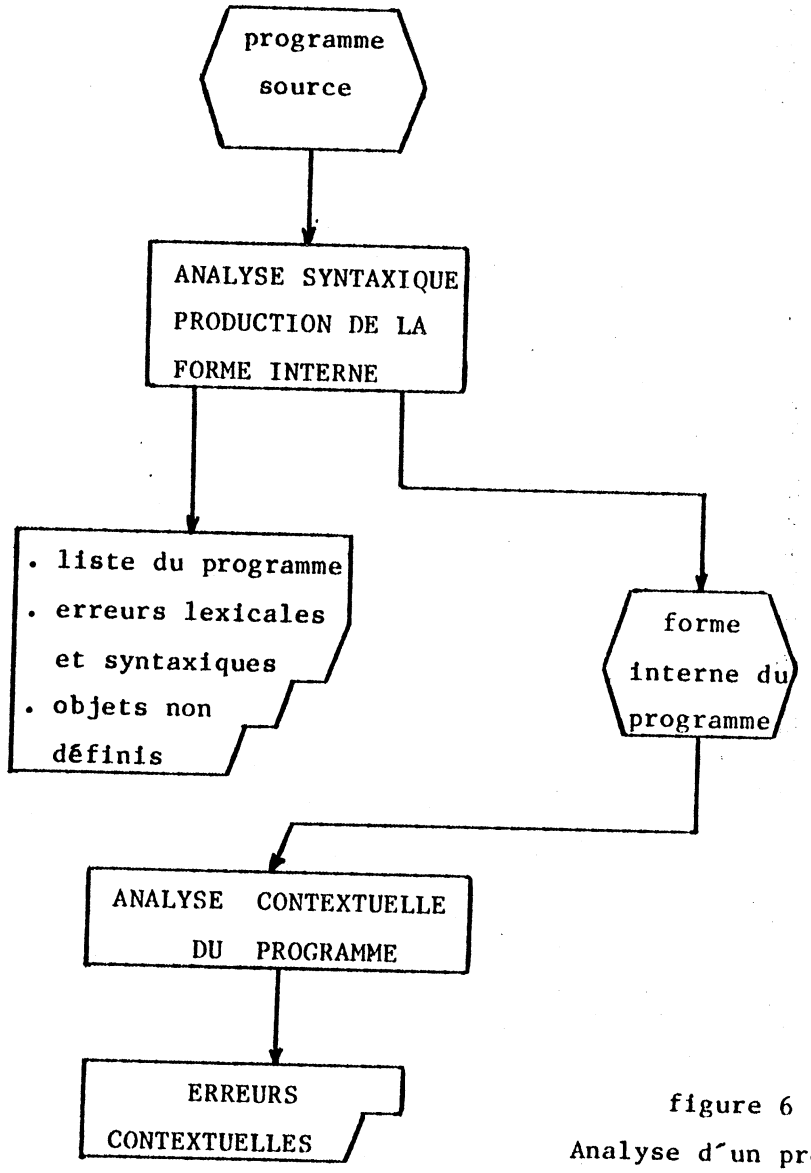


figure 6 :
Analyse d'un programme

4.2.2 - Structure de la forme interne du programme

Nous venons de voir que la forme interne est l'élément charnière de l'environnement de production (cf §4.1). Elle comporte donc les informations nécessaires
- aux vérifications contextuelles,
- à la production de code BASIC,
- à l'édition de la liste du programme MEFIA et de l'index par le documenteur.

Elle joue également un rôle important du point de vue de la portabilité du système : on peut par exemple vouloir implanter des composants de l'environnement sur des sites différents ; pour cela, une structure standard des fichiers contenant la forme interne est nécessaire.

Contenu de la forme interne :

La forme interne est constituée par les éléments suivants :
- le programme sous une forme proche de la forme externe, auquel sont attachées des informations complémentaires (attributs),
- le fichier des commentaires, utilisé par le documenteur,
- les tables annexes qui regroupent toutes les informations concernant les objets définis dans le programme,
- la liste des références externes du programme source, c'est-à-dire les primitives MEFIA-EAO utilisées.

a) Le programme

La forme interne du programme suit une grammaire proche mais simplifiée de la notation MEFIA. Les expressions y sont sous la forme post-fixée ; des éléments facultatifs au niveau de la notation ont été ajoutés systématiquement pour simplifier la grammaire. A chaque unité syntaxique sont greffés les attributs, c'est-à-dire les informations complémentaires nécessaires aux autres composants de l'environnement de production.

Ces informations complémentaires sont essentiellement :
- le numéro du symbole terminal (résultat de l'analyse lexicale),
- la valeur du symbole,
- un accès dans la structure annexe qui contient les informations complémentaires,
- le numéro de ligne dans le programme source où apparaît le symbole concerné.

b) Les commentaires

Afin d'alléger la forme interne, les commentaires sont rangés dans un fichier annexe et remplacés dans la forme interne par un symbole signalant la présence d'un commentaire et son accès dans le fichier annexe. Ceci est suffisant pour permettre l'édition du programme par le documenteur.

c) Les tables annexes

Les tables annexes regroupent les informations concernant chaque identificateur du programme, élaborées au premier passage:
- le type de l'information nommée,
- l'environnement de sa définition (module et procédure où elle est définie),
- son adresse d'implantation dans le programme produit par le traducteur,
- d'autres informations qui varient en fonction du type de l'objet.

d) Liste des primitives MEFIA-EAO utilisées :

La liste des primitives MEFIA-EAO utilisées constitue un ensemble de références à des procédures de la bibliothèque gérée par l'environnement. Les informations nécessaires à l'édition de liens sont incluses par le traducteur dans le programme objet non exécutable.

4.2.3 - Premier passage : analyse lexicographique et syntaxique

La réalisation des composants de l'environnement, se fait par la mise en oeuvre de méthodes, de techniques et d'outils connus et maîtrisés. Cette attitude est systématique pour toute réalisation dans le projet MOSAIQUE.

L'analyseur syntaxique est produit à partir de la grammaire du langage d'entrée. On utilise pour cela un outil de développement spécialisé [PeG 68] [DeG 81]. Il permet à partir de la grammaire donnée sous forme de règles hors contexte :

- de vérifier qu'elle satisfait les conditions LL(1) (ce qui implique qu'elle n'est pas ambiguë)
- d'optimiser les règles par des transformations systématiques
- de produire le programme d'analyse syntaxique correspondant : ce programme est produit soit dans le langage de programmation spécifié par l'utilisateur (PL/1, FORTRAN, etc...), soit sous la forme d'instructions codées, à interpréter.

Cette dernière forme a été adoptée : elle permet, au niveau de la réalisation de l'analyseur :

- une **indépendance du programme d'analyse** par rapport à une machine ou un langage de programmation.

- un **gain en place mémoire** occupée par le programme : les valeurs des codes produits sont étudiées pour être représentées sur des mots de 16 bits ; d'autre part, l'interpréteur de ces codes est très simple et occupe une place mémoire réduite.

- une grande **souplesse dans la mise au point** de l'analyseur : en cas de modification de la grammaire il suffit de produire la nouvelle version des tables d'analyse et de remplacer l'ancienne version par la nouvelle pour obtenir l'analyseur syntaxique modifié. Aucune programmation n'est nécessaire ; seule la grammaire est à modifier ; ceci est toujours vrai lorsque l'on produit l'analyseur à l'aide d'un outil automatique.

L'analyse lexicographique, qui consiste en l'acquisition d'un symbole terminal, est directement programmée à partir du graphe d'analyse lexicale. Une autre technique consisterait à utiliser le même outil syntaxique LL(1) pour réaliser la lexicographie. Ce choix n'est pas fait pour des raisons d'encombrement mémoire : la taille du code produit par l'outil pour l'analyse lexicographique est importante par rapport au programme correspondant au graphe, transcrit directement. Ceci vient du fait qu'à ce type d'analyse correspond un automate d'états finis, alors qu'à l'analyse syntaxique correspond un automate à pile.

Les actions de compilation, souvent appelées actions sémantiques (voir remarque §4.2.1), sont des actions à exécuter entre l'acquisition de deux symboles terminaux. Elles sont insérées dans la grammaire et sont traduites dans le code produit par un appel de l'action de compilation spécifiée.

En résumé, les étapes suivies pour la réalisation de l'analyseur syntaxique ont été les suivantes :

a - Description de la syntaxe du langage en entrée sous la forme d'une grammaire LL(1).

b - Insertion dans la grammaire des actions de compilation à exécuter.

c - Production de l'analyseur syntaxique sous la forme d'instructions codées à interpréter, à partir d'un outil spécialisé.

d - Ecriture de l'interpréteur du code produit.

e - Description des actions de compilation à exécuter au cours de l'analyse : actions de conservation d'informations, de production de la forme interne et de signalisation des erreurs rencontrées.

L'interpréteur du code produit par l'outil syntaxique LL(1) reste le même quelle que soit la syntaxe d'entrée. En conséquence, cet

interpréteur a pú être utilisé pour la construction des autres composants de l'environnement dont le texte d'entrée fait l'objet d'une analyse syntaxique.

C'est le cas des composants du système de production, travaillant à partir de la forme interne :
- le deuxième passage de l'analyseur,
- le documenteur,
- le traducteur.

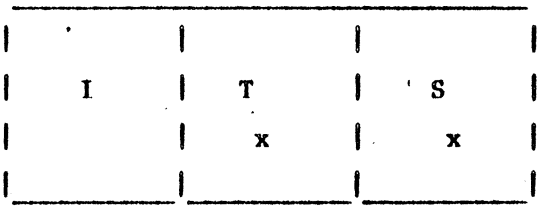
Les étapes de construction de ces outils ont donc été les suivantes :

a - Description de la grammaire de la forme interne sous une forme LL(1), en y spécifiant les actions de compilation à exécuter.

b - Production du programme correspondant (tables) à l'aide de l'outil syntaxique LL(1).

c - Ecriture des actions de compilation qui réalisent les opérations spécifiques à l'outil construit ; c'est la seule partie de programme à écrire dans la construction d'un composant.

Il en découle la structure générale suivante des outils construits:



I : interpréteur des tables d'analyse syntaxique (écrit une seule fois)

T : tables d'analyse syntaxique du composant x (obtenu
x automatiquement à partir du transformateur de grammaires).

S : actions de compilation à exécuter pendant l'acquisisition
x du texte d'entrée.

Langage d'écriture : Comme pour le didacticiel, le langage utilisé pour la description des actions de compilation a été la notation MEFIA ; la traduction manuelle systématique des actions spécifiées a été effectuée vers BASIC pour les raisons suivantes :

- cela permet de bénéficier de l'expérience acquise dans l'utilisation de BASIC,

- pour l'installation du système de production sur la machine, il suffit d'avoir le langage BASIC disponible pour exécuter à la fois les outils et les programmes produits. On a ainsi la possibilité de tester des programmes produits sur le site de production, avant leur transfert sur le site de diffusion.

4.2.4 - Deuxième passage : les vérifications contextuelles

La phase de vérifications contextuelles permet la détection des erreurs de cohérence du programme, ainsi que celle des erreurs qui risquent de provoquer des effets de bord à l'exécution. Nous donnons ici les principales vérifications effectuées au cours de ce passage.

4.2.4.1 - Le mélange de types

D'une manière générale, MEFIA interdit le mélange de types (conversions implicites) dans une affectation, une expression ou un passage de paramètres. En fait, la seule conversion implicite tolérée est celle d'un entier en réel. Pour toute autre combinaison, la conversion doit se faire explicitement.

4.2.4.2 - L'affectation de n-uplets

On trouve dans [Mor 79] une étude de ce problème. Seule l'affectation d'un n-uplet à un autre est autorisée, à condition que le n-uplet spécifié en partie droite ne contienne pas d'appel à une action de modification à résultats ; ceci permet d'éviter d'éventuels effets de bord.

Exemple : Soit A1 une action à résultat modifiant la variable globale B de type t0 et produisant un résultat de type t1. Le résultat de l'instruction suivante n'est pas défini :

[X,Z] <- [A1,B]

{ X est de type t1 et Z de type t0 }

Suivant l'ordre d'exécution de la liste d'expressions spécifiée en partie gauche de l'affectation, la valeur rangée dans Z sera différente.

Pour éviter ce problème, la notation MEFIA n'autorise l'utilisation d'une action de modification à résultats que seule en partie droite d'une affectation. Pour des raisons de clarté des algorithmes décrits en MEFIA, cette restriction est également appliquée aux autres procédures à plusieurs résultats : fonctions et opérateurs (cf §3.3.2).

4.2.4.3 - Les autres vérifications

Les autres vérifications faites au cours de ce passage sont les suivantes :

- Accessibilité des variables ou des procédures utilisées, en fonction du contexte d'exécution : droit ou non à l'utilisation d'une variable ou d'une procédure. On assure ici la protection des objets locaux à un bloc (univers, procédures).

- Cohérence du type des paramètres dans un appel de procédure.

- Cohérence du type des résultats d'une procédure à l'envoi et à la récupération des résultats.

- Correspondance entre un tableau et l'intervalle qui lui est lié, lors du passage d'un tableau en paramètre.

- Absence de synonymies d'objets, c'est à dire vérification que tout objet a un nom unique dans tout contexte d'exécution.

Remarque : Un certain nombre de vérifications ne peuvent être faites qu'à l'exécution :
- vérification qu'une variable suit la propriété avec laquelle elle est définie,
- vérifications de débordement d'intervalle à l'indexation de tableaux et à la modification d'indices,
- etc...

Ces vérifications sont assurées par le code produit par le traducteur.

La réalisation du programme effectuant ce deuxième passage se fait en appliquant la démarche présentée dans la partie précédente (cf §4.2.3). Les actions de vérification et de signalisation des erreurs rencontrées constituent la seule partie de programme à écrire explicitement.

4.3 - Le documenteur de programmes

Le documenteur permet à l'utilisateur d'obtenir des documents concernant son programme MEDIA-EAO. Les documents sont produits à partir de la forme interne.

Données : Forme interne du programme :
- programme interne
- tables annexes
- fichier des commentaires.

Résultats : Documents concernant le programme :
- liste présentée du programme
- dossier
- index.

La méthode mise en oeuvre pour la production du documenteur est la même que pour réaliser les autres composants travaillant à partir de la forme interne (cf §4.2.3).

Nous décrivons dans cette partie les documents produits par le documenteur.

4.3.1 - La liste du programme MEFIA-FAO

Le programme MEFIA-FAO est édité en tenant compte de la syntaxe :

- Les mots-clés de la notation sont soulignés.

- Les instructions sont décalées en fonction des structures de contrôle, des procédures et des univers (modules) ; chaque ligne de programme est numérotée.

- En fonction de sa position dans le programme, un commentaire est édité soit entre deux instructions, soit en regard de l'instruction courante. On donne de plus la possibilité à l'utilisateur de forcer l'édition par une commande placée en tête du commentaire. Cette commande permet l'édition de ce commentaire dans le programme entre deux instructions, en regard d'une instruction ou en fin de bloc, avec simplement une référence à ce commentaire dans le texte du programme.

4.3.2 - Le dossier

Le dossier correspond à l'édition des structures de bloc du programme (univers, procédures), en ne faisant apparaître pour chaque bloc que le commentaire principal placé en tête.

Ce commentaire doit contenir la spécification du bloc, et son rôle dans le programme. L'obtention de ce document permet à l'utilisateur d'avoir à sa disposition l'équivalent d'un dossier de programmation, dans la mesure où les commentaires sont suffisamment explicites.

4.3.3 - L'index ou table des références croisées

L'index [Sch 78.a] fournit toutes les informations données habituellement par les compilateurs dans la table des références croisées, c'est à dire :

- le nom de l'objet,
- son type,
- le numéro de la ligne de programme où il est défini,
- la liste des lignes où l'objet est utilisé, avec marquage des lignes où il est modifié,
- son adresse interne dans le programme produit par le traducteur, c'est-à-dire son adresse relative ou absolue dans le système à l'exécution.

Les références (numéros de lignes) sont données soit par rapport à la liste du programme produite par le documenteur, soit par rapport au texte source donné par l'utilisateur. Ceci permet à l'utilisateur de travailler, s'il le désire, à partir de sa propre présentation du programme.

L'ensemble de ces documents facilite à l'utilisateur la documentation de son programme ainsi que sa mise au point. La production de chaque document est optionnelle ; un dialogue permet à l'utilisateur de spécifier les documents souhaités.

4.4 - Le traducteur

Le traducteur produit, à partir de la forme interne, le programme BASIC correspondant. La traduction se fait en deux passages :

- Premier passage : la production du programme objet en BASIC standard

  Données : Forme interne du programme :
  - programme interne
  - tables annexes
  - liste des primitives MEFIA-FAO

  Résultats : Programme objet :
  - programme objet en BASIC
  - liste des références externes
  - liste des références en avant

- Deuxième passage : L'optimisation du code produit (facultatif)

  Données : - Programme objet en BASIC standard non exécutable
  - Liste des références externes

  Résultats : - Programme objet optimisé en BASIC standard non exécutable
  - Liste mise à jour des références externes

Pour produire le code objet, il est tenu compte des limitations du langage BASIC (cf §3.6.1). Seules quelques instructions standards sont utilisées. Le programme produit n'est pas "lisible" ; le système à l'exécution ainsi défini est transparent à l'utilisateur et aucun commentaire n'est produit.

. le premier passage a pour fonction de produire, à partir de la forme interne, le programme objet en BASIC. La méthode appliquée pour la réalisation de ce composant est la même que pour le documenteur (cf §4.2.3).

. **le second passage** réalise un certain nombre d'améliorations du code produit en tenant compte des caractéristiques particulières des programmes traduits. Ces optimisations sont faites directement à partir du BASIC produit au premier passage.

4.4.1 - Le système à l'exécution

Afin d'éviter les problèmes dûs à la limitation du nombre d'identificateurs en BASIC et notamment la gestion de ceux-ci, et aussi afin de pouvoir appliquer facilement les techniques classiques de compilation, on choisit de regrouper tous les objets du programme utilisateur dans une zone mémoire structurée en pile. Les seuls types de variables manipulables en BASIC sont les **réels** (représentés en général sur 4 à 8 octets) et les **caractères** (1 octet par caractère). Très souvent aussi, on peut manipuler des **entiers** représentés sur deux octets. Il n'y a pas de type logique en BASIC.

La pile des objets de l'utilisateur est représentée en BASIC par trois piles distinctes : une pile d'entiers qui contient les informations entières et logiques, une pile de réels et une pile de caractères (figure 7).

Remarque : Un autre choix possible est de n'avoir qu'une pile de réels dans laquelle sont regroupés les objets entiers, réels et logiques. Cette solution est écartée, car la majorité des informations manipulées est formée d'entiers représentables sur 16 bits. Les représenter par le double ou le quadruple de place mémoire nécessaire provoque une trop grande perte de place. Le choix fait permet un gain en place mémoire et en temps de calcul, la manipulation d'entiers étant moins coûteuse que celle des réels.

Les piles sont structurées de façon identique ; elles sont composées :

- d'une partie statique contenant tous les objets statiques du programme, c'est-à-dire les informations globales à chaque univers et au programme principal.

- d'une partie dynamique qui contient les environnements des procédures actives, c'est-à-dire les objets locaux des procédures exécutées à un instant donné. Cet espace mémoire est alloué aux objets à l'exécution du programme utilisateur.

Remarque : La gestion d'un tas [CGV 80] à l'exécution n'est pas nécessaire : en effet, MEFIA ne gère pas d'objets de type "pointeur", et n'a pas d'instructions d'allocation dynamique de mémoire.

Représentation des informations dans les piles (figures 7 et 8)

. Les variables simples : entiers, réels et caractères sont représentés respectivement par un élément de la pile d'entiers, de réels ou de caractères. Une valeur logique est représentée par un élément de la pile d'entiers ne pouvant prendre que les valeurs 1 ou -1.

. Les tableaux : les éléments des tableaux sont représentés de manière contiguë dans la pile correspondant à leur type. L'origine virtuelle [CGV 80] est une adresse dans cette pile ; c'est donc une valeur entière rangée dans la pile des entiers.
On ne connaît en général la taille d'un tableau qu'à l'exécution. La place des éléments est réservée (allouée) à la suite des objets simples à l'exécution.

. Les intervalles : les intervalles sont représentés par un triplet d'entiers contenant la borne inférieure, la borne supérieure et le cardinal de l'intervalle. Ces valeurs sont calculées à l'exécution.

. **Les paramètres** : un paramètre ne peut être modifié en MEFIA afin d'éviter des effets de bord ; ceci est verifié par l'analyseur. Le passage des paramètres se fait par référence : les adresses des paramètres sont rangées dans la pile des entiers, à la base de l'environnement créé à l'appel de la procédure.

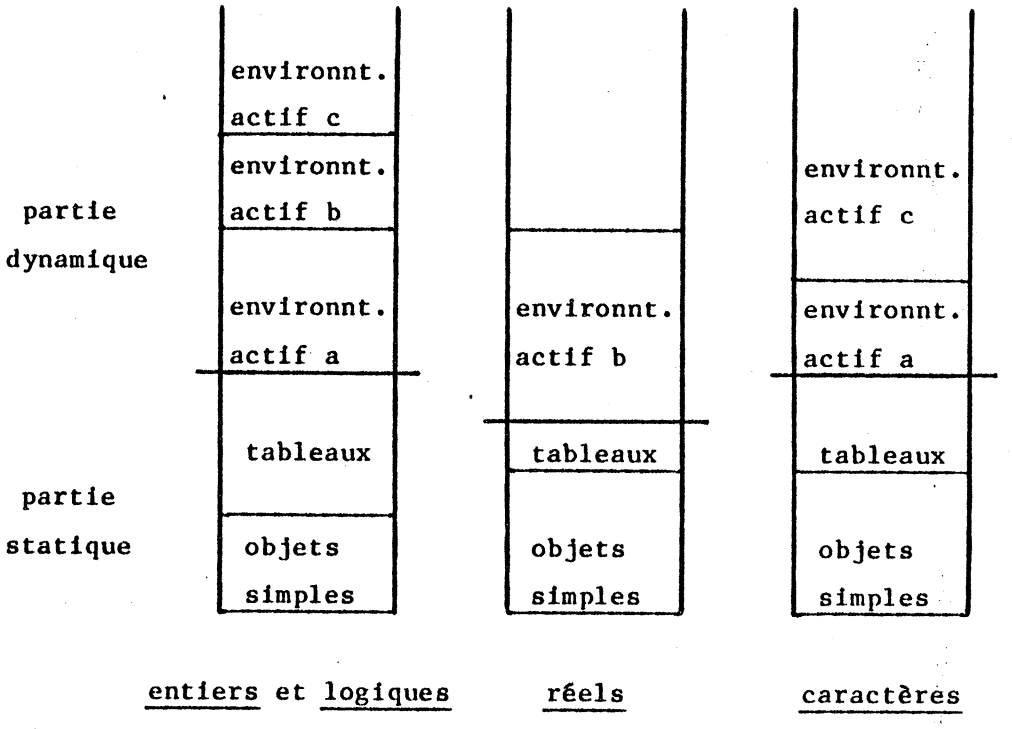


figure 7 : Représentation de la pile à l'exécution.

Dans la représentation de la figure 7, l'environnement b ne comporte aucun objet de type caractère et les environnements a et c, aucun objet de type réel.

Les environnements sont créés dans l'ordre d'activation des procédures auxquelles ils correspondent. Ils sont chaînés entre eux par des liens dynamiques pour assurer le retour à l'environnement appelant (figure 8).

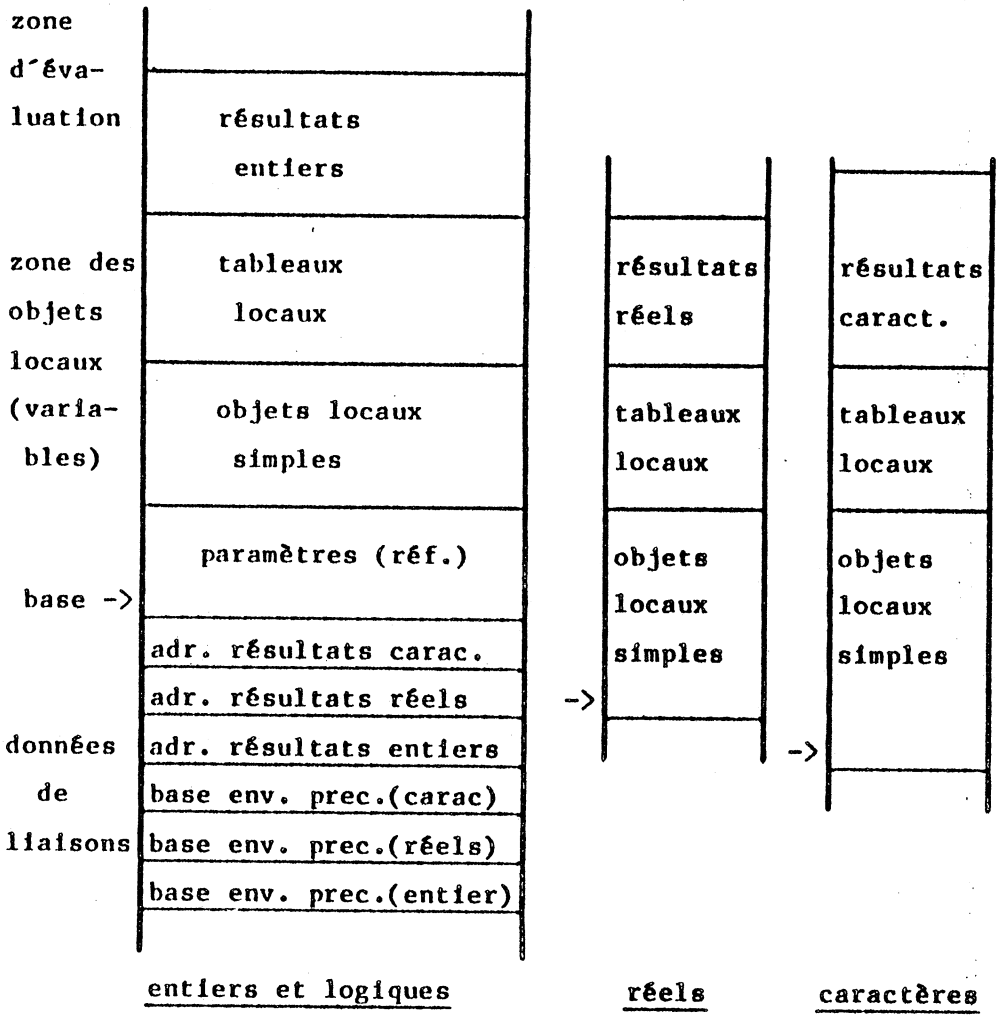


figure 8 : Représentation d'un environnement.

Cette organisation, proche de celle adoptée dans les compilateurs classiques, permet de gérer les appels récursifs de procédures et permet une bonne gestion de la mémoire.

4.4.2 - Structure du programme produit

4.4.2.1 - Le code produit

Le code produit est formé des instructions BASIC standards de base mises en évidence pour la production manuelle (cf §3.5.3). Les objets manipulés par le code produit sont les informations rangées dans les piles du système à l'exécution.

Les déclarations d'objets comportant des parties dynamiques sont traduites par des instructions d'allocation et d'initialisation, d'éléments de ces piles.

Les procédures sont traduites par des "subroutines BASIC". Les règles de traduction des instructions sont proches de celles définies pour la traduction manuelle ; l'évaluation des expressions est faite opération par opération, à partir de leur représentation postfixée contenue dans la forme interne. Cette décomposition est nécessaire pour traiter le cas des expressions contenant des appels de fonctions : les appels à ces procédures doivent se faire au cours de l'évaluation et ne peuvent se traduire que par un appel de subroutine.

Les **références en avant** dans le programme produit sont calculées au fur et à mesure de la production ; ces valeurs sont transmises à l'éditeur de liens pour être insérées dans le code produit.

Dans certains cas, le code produit comporte des références à des **procédures annexes**, spécifiques à la traduction, pour diminuer la taille du code produit. Par exemple, pour produire des vérifications dynamiques, on engendre le code réalisant l'appel de la procédure de vérification. Le traducteur établit la liste des procédures annexes utilisées pour la transmettre à l'éditeur de liens. Il complète ainsi la liste des références externes du programme.

Comme pour la traduction manuelle, les appels de primitives MEFIA-EAO sont traduits par des appels à des subroutines de la bibliothèque. Chaque appel constitue une référence à un objet externe du programme. Il est représenté dans le code produit par un numéro de ligne fictif. A l'édition de liens, la primitive utilisée est concaténée au programme, le numéro de ligne BASIC correspondant à son point d'entrée constitue la valeur de la référence résolue. La liste des primitives utilisées par le programme est élaborée au cours de l'analyse (cf §4.2.1). Les conventions de liaison (passage de paramètres, récupération de résultats) sont les mêmes que pour les autres procédures du programme.

Les paramètres et les variables locales des primitives sont des entités BASIC externes aux piles du système à l'exécution. Ceci vient du fait que les primitives sont réalisées manuellement. Une interface entre le programme produit et les procédures de la bibliothèque est nécessaire : il s'agit de procédures (subroutines) qui, pour chaque primitive, assurent la transmission des paramètres dans les variables BASIC correspondantes de la primitive, et le rangement des résultats, s'il y a lieu, dans la pile à l'exécution. La séquence d'appel d'une primitive peut être simplifiée en phase d'optimisation.

L'ensemble des informations gérées à l'exécution est donc formé des trois piles et des variables manipulées par les primitives MEFIA-EAO.

4.4.2.2 - Organisation du programme produit

Le programme est organisé de façon à assurer que toutes les instructions correspondant à des déclarations statiques soient exécutées avant la première instruction du programme principal. L'ordre de production des instructions de déclaration et des procédures correspond à leur ordre physique dans le programme source (figure 9). Afin qu'au début ne soient exécutées que les instructions correspondant aux déclarations des objets statiques, on lie ces séquences d'instructions les unes aux autres par des instructions de branchement inconditionnel.

Remarque : Chaque branchement à la séquence d'instructions suivante constitue une référence en avant dans le programme produit.

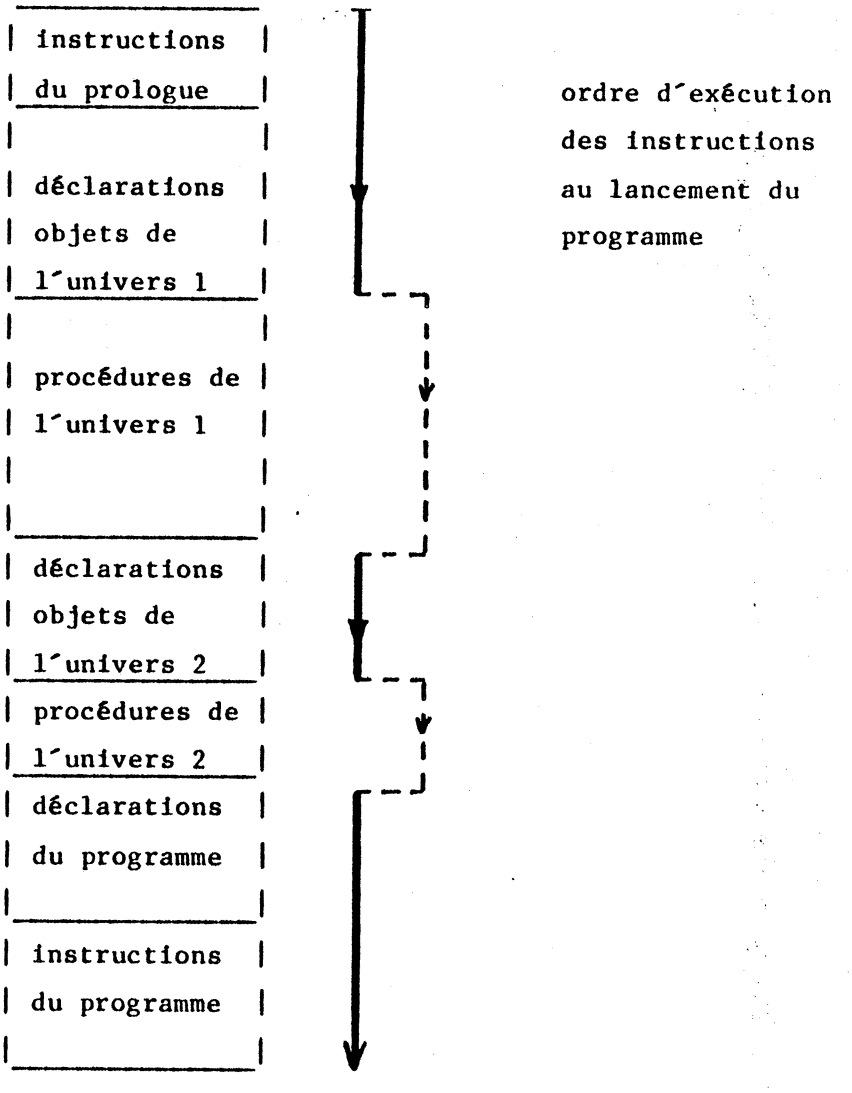


figure 9 : Organisation du programme produit.

Dans la notation MEFIA, le "prologue" regroupe les déclarations des constantes d'exécution globales à tout le programme. Ce bloc particulier n'est pas utilisé dans la description du didacticiel.

Le programme objet se présente de la façon suivante :

- un programme BASIC non exécutable contenant des références en avant et des références externes à résoudre à l'édition de liens
- un ensemble de références externes à résoudre
- un ensemble de références en avant à résoudre.

4.4.2.3 - Les insuffisances du code produit

Le principal intérêt du code produit réside dans le fait qu'il est indépendant de toute machine et de toute version en BASIC. D'autre part, la gestion dynamique des environnements permet la programmation de procédures récursives.

Le code produit présente toutefois un certain nombre de défauts et n'est pas optimisé. Les règles de traduction appliquées sont indépendantes des caractéristiques du programme source ; on ne tient compte d'aucun cas particulier. Ceci ressort particulièrement au niveau de l'évaluation des expressions et de l'appel des primitives prédéfinies :

- les expressions sont systématiquement décomposées et évaluées opération par opération, alors que BASIC peut traiter directement la plupart des expressions. La décomposition ne s'avère nécessaire que lorsque l'expression contient des appels de procédures, ou que les opérandes ne sont pas numériques.
- la réalisation d'un appel de primitive est coûteuse : le passage de paramètres devrait se faire par valeurs, directement dans les variables BASIC que manipule la primitive, plutôt que de charger les références dans la pile et de passer par une interface.

Un autre défaut du code produit est dû à la gestion dynamique des environnements : chaque objet du programme utilisateur est un élément de tableau, ce qui représente un coût d'accès important par rapport à une variable simple et augmente la taille du code engendré.

La description d'un dialogue ou d'une situation pédagogique étant essentiellement faite en termes de primitives MEFIA-EAO, la taille du code produit est disproportionnée par rapport à celle du programme source, et il se pose des problèmes d'implantation du code produit sur la machine cible.
Le code produit pour l'appel peut facilement être réduit, si l'on connaît à la traduction **les noms des objets BASIC dans lesquels doivent être transmises les valeurs des paramètres.**

4.4.3 - Optimisation du code engendré

Nous venons de mettre en évidence les défauts du code BASIC engendré, notamment la taille trop importante de celui-ci. Un passage est effectué sur le code produit , afin d'en réduire la taille. Les principales transformations effectuées tiennent compte des caractéristiques particulières des programmes à traduire:

- ils comportent de nombreux appels à des primitives prédéfinies,
- les expressions comportent assez rarement des appels à des fonctions,
- aucun objet manipulé n'est de type réel, du fait que le logiciel produit est un didacticiel de langue.

A partir de ces constatations, les optimisations suivantes sont réalisées :

- simplification des appels de primitives prédéfinies,
- réduction, lorsque c'est possible, du code d'évaluation des expressions,
- suppression des instructions qui gèrent la pile des objets de type réel, lorsque le programme traduit ne comporte aucun objet de ce type,

. réalisation de certaines optimisations locales simples, par reconnaissance et simplification de certains schémas d'instructions.

4.4.3.1 - Optimisation des séquences d'appels de primitives

Pour réduire la séquence d'appel d'une primitive, il faut connaître les noms des objets BASIC dans lesquels doivent être transmises les valeurs des paramètres, et de ceux dans lesquels sont rangés les résultats produits.

Ces informations sont contenues dans la bibliothèque de primitives : on trouve pour chaque primitive, les noms des variables BASIC correspondant à chaque paramètre et à chaque résultat produit. L'outil qui assure la gestion de la bibliothèque met à jour ces informations à chaque modification.

La séquence de code correspondant au passage des paramètres est remplacée par des affectations des valeurs des paramètres aux variables BASIC correspondantes. L'opération symétrique est réalisée pour la récupération des résultats : par des affectations, on range dans la pile les résultats produits.

On réalise ainsi directement les opérations faites précédemment par l'interface entre la pile à l'exécution et les primitives. Celui-ci devient désormais inutile.

4.4.3.2 - Optimisation des expressions

La simplification des expressions est réalisée par la reconnaissance de la séquence de code (schéma) correspondant à l'évaluation d'une opération binaire, et à son remplacement par l'expression parenthésée équivalente. On recrée ainsi l'expression totalement parenthésée à partir de son évaluation.

Le gain en temps d'exécution est minime, mais la taille du code engendré est très nettement réduite : approximativement divisée par cinq.

4.4.3.3 - Autres optimisations

Un certain nombre d'autres optimisations sont réalisées :

- Si le programme utilisateur ne manipule aucun objet de type réel ou de type caractère, toutes les instructions qui gèrent la (les) pile(s) correspondante(s), engendrées systématiquement en phase de traduction, sont supprimées.

- Des optimisations locales simples sont assurées. Elles correspondent à des cas particuliers de code produit. Par exemple, on supprime l'empilement d'un objet suivi du dépilement de celui-ci. Après reconnaissance du cas particulier, on produit la séquence de code simplifiée.

Le choix de faire un passage sur le code produit plutôt que de réaliser ces optimisations dès la phase de traduction permet d'éviter une surcharge de travail au traducteur, et donc une taille trop importante de celui-ci. De plus, ceci simplifie sa réalisation et évite des problèmes d'implantation.

L'ensemble des optimisations réalisées permet de réduire la taille du code engendré de 60 à 70% dans certains cas. Ce résultat est essentiellement dû aux caractéristiques très particulières des programmes traduits.

L'utilisation de cet outil est facultative, c'est-à-dire que l'utilisateur peut sauter cette phase dans la production, et obtenir un programme exécutable à partir du code produit en phase de traduction. dans ce cas, une taille mémoire importante est nécessaire pour l'exécution.

4.4.4 - Conclusion

Nous venons de définir un système permettant d'obtenir un programme BASIC **indépendant de la machine cible,** à partir de sa forme interne. Les méthodes et les techniques appliquées pour sa réalisation sont des méthodes classiques de compilation [GOH 74] [CSV 76].
Le code engendré n'est pas exécutable. Il comporte des références externes et des références en avant à résoudre. L'éditeur de liens rend le programme exécutable en résolvant les références et en concaténant les primitives de la bibliothèque utilisées, au programme.

L'adaptation du programme à son site d'exécution se fait par l'utilisation de la version de la bibliothèque qui correspond à la machine cible. Pour assurer la diffusion, il est donc nécessaire de gérer autant de versions de la bibliothèque qu'il y a de dialectes de BASIC disponibles sur les sites d'exécution. Cette gestion est facilitée par une organisation de la bibliothèque qui met bien en évidence sa partie non portable.

4.5 - L'éditeur de liens

L'éditeur de liens a pour fonction de transformer le programme objet produit par le traducteur en un programme exécutable. Pour cela, il doit :

- résoudre les références externes,
- résoudre les références en avant.

Il assure la portabilité du logiciel produit en l'adaptant à la machine cible spécifiée par l'utilisateur. Il concatène au programme objet les primitives de la bibliothèque utilisées (figure 10). Pour cela, il doit avoir accès à la version des primitives, adaptée à cette machine.

données : - Programme objet fourni par le traducteur :
  . programme objet en BASIC standard non exécutable. (éventuellement optimisé)
  . liste des références externes
  . liste des références en avant.
- Bibliothèque des primitives.
- Nom de la machine cible.

résultats : Programme BASIC exécutable sur le matériel spécifié.

Un programme BASIC exécutable est constitué d'une suite de lignes numérotées en ordre croissant. Chaque référence non résolue doit donc être transformée en un numéro de ligne dans le programme exécutable.

4.5.1 - Résolution des références en avant

Une référence en avant correspond à une ligne de programme dont on ne connaît pas le numéro au moment de la traduction : elle désigne une séquence de code non encore produite. Elle est représentée dans le code objet par un numéro de ligne fictif. Le numéro de ligne effectif correspondant est déterminé au cours de la production, lors de la traduction de la séquence de programme désignée par la référence.

En fin de traduction, les numéros de lignes correspondant aux références en avant sont calculés et transmis à l'éditeur de liens.

L'opération de résolution des références en avant à effectuer par l'éditeur de liens consiste donc simplement à remplacer dans le programme objet chaque numéro de ligne fictif par sa valeur effective.

4.5.2 - Résolution des références externes

Les références externes à résoudre correspondent
- soit à des primitives MEFIA-EAO utilisées dans le programme,
- soit à des procédures annexes utilisées pour réduire le code produit à la traduction (cf. §4.4.2.1).

Le code de ces primitives est contenu dans la bibliothèque gérée par le système (cf §4.6). Certaines primitives font référence à des procédures auxiliaires, notamment à celles qui constituent la partie spécifique à une machine cible donnée. Ces appels constituent des références externes supplémentaires à résoudre également par l'éditeur de liens.

La liste des références externes fournie par le traducteur est donc incomplète : elle ne comporte pas ces primitives auxiliaires. Pour compléter cette liste, on utilise le graphe des appels des procédures dont est constituée la bibliothèque. Ce graphe est géré par les outils assurant la maintenance de la bibliothèque (cf. §4.6.4). Il décrit, pour chaque procédure contenue dans la bibliothèque, la liste des procédures qu'elle appelle.

Pour obtenir la liste de toutes les procédures nécessaires à l'exécution de chaque primitive, il faut calculer la fermeture transitive de ce graphe. Ce calcul est effectué à chaque mise à jour de la bibliothèque.

L'éditeur de liens complète la liste des références externes à partir de cette fermeture transitive.

A partir de cette liste complétée, les opérations suivantes sont effectuées :

- calcul des numéros de lignes effectifs de chaque procédure externe utilisée

- remplacement, dans le programme objet, des numéros de lignes fictifs par les valeurs trouvées

- concaténation des procédures externes au programme objet.

Remarque : Suivant la version de BASIC disponible, un ensemble d'instructions non standards doit se trouver au début du programme exécutable. Un en-tête de programme, adapté à chaque machine cible, est conservé dans la bibliothèque et placé par l'éditeur de liens en tête du programme exécutable.

Le programme objet est formé de lignes numérotées en ordre croissant de un en un. Les primitives concaténées sont numérotées de la même manière, à la suite : le code BASIC est renuméroté avant la concaténation. Cette opération offre deux intérêts principaux :

- Elle permet de gérer une procédure simplement, sans avoir à se préoccuper de ses numéros de lignes par rapport aux autres procédures; une primitive peut être réalisée ou modifiée sans containte de numérotation.

- Elle permet à l'éditeur de liens de calculer les numéros de lignes des procédures dans le programme exécutable, à partir de la taille, en lignes BASIC, du code de chaque procédure. Ces tailles sont des informations contenues dans la bibliothèque.

Ces informations sont suffisantes car la première ligne de chaque primitive constitue l'adresse BASIC de son point d'entrée.

En résumé, l'éditeur de liens effectue successivement les opérations suivantes :

a - le calcul de toutes les primitives nécessaires à l'exécution du programme

b - le calcul, pour chaque primitive utilisée, de son numéro de ligne effectif dans le programme exécutable

c - la résolution des références dans le programme objet:
- .résolution des références en avant
- .résolution des références externes

d - le placement de l'en-tête adapté à la machine cible au début du programme exécutable

e - la concaténation de chaque primitive utilisée, en renumérotant ses lignes et en résolvant ses références externes.

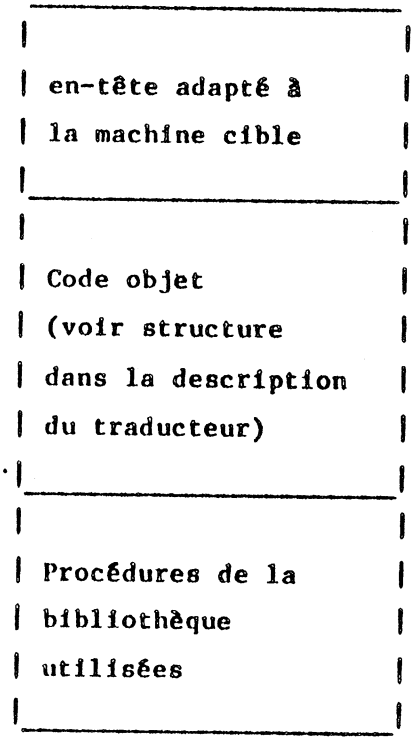


figure 10 : Structure du programme exécutable.

L'adaptation à la machine cible est assurée par l'utilisation de la bibliothèque dont le noyau est écrit dans la version de BASIC disponible sur cette machine.

4.6 - La bibliothèque de primitives MEDIA-FAO

4.6.1 - Fonctionnalités

La bibliothèque regroupe toutes les procédures nécessaires à l'exécution d'un programme objet donné. Dans la présentation des différents composants du système, nous avons montré son rôle central. En effet, de nombreux outils travaillent à partir d'informations dont elle assure la gestion :

a - **l'analyseur** consulte, pour chaque primitive MEDIA-FAO :

- le nom de la primitive, sa nature (action, fonction, modification),
- le nombre et le type de ses paramètres et de ses résultats, pour vérifier la correction de son appel dans le programme source et produire les informations correspondantes dans la forme interne.

b - **le traducteur** fait référence à des procédures annexes nécessaires à l'exécution du code produit. La phase d'**optimisation** consulte, pour chaque primitive MEDIA-FAO utilisée dans le programme, les noms des objets BASIC par lesquels se fait la transmission des paramètres et des résultats. Elle fait référence à des procédures annexes complémentaires qui optimisent certaines parties de code.

c - **l'éditeur de liens** utilise (cf. §4.5) :

- la fermeture transitive du graphe des appels des procédures gérées,
- le code des procédures appelées, dans la version de BASIC disponible sur la machine cible,
- la taille, en nombre de lignes, de chaque procédure,
- l'en-tête de programme adapté à la machine cible.

L'ensemble de ces informations doit être géré de sorte que des modifications sur leur contenu puissent être assurées facilement.

Après avoir décrit l'organisation des procédures gérées, nous présentons les outils construits pour assurer cette gestion et la cohérence des informations.

4.6.2 - Organisation des procédures de la bibliothèque

Nous trouvons deux catégories de procédures dans la bibliothèque :

- Les procédures "nommées" qui correspondent à des primitives MEFIA-EAO

- Les procédures "non nommées" ou "anonymes" qui correspondent :
  - aux procédures auxiliaires nécessaires à l'exécution des procédures "nommées",
  - aux procédures annexes au code produit par le traducteur.

L'ensemble de ces procédures est organisé par couches. Chaque couche correspond à un niveau de spécification. Chaque procédure est décrite en BASIC standard, en termes de procédures de niveau inférieur. Seul le niveau le plus bas est décrit dans la version de BASIC spécifique à une machine donnée. Il constitue la partie non portable de la bibliothèque.

Les couches de procédures mises en évidence :

Nous présentons les niveaux mis en évidence, du plus spécifique au plus général.

Remarque : Les procédures annexes au code objet du traducteur constituent un sous-ensemble de la bibliothèque, séparé des autres procédures. Elles sont écrites en BASIC standard et n'apparaissent pas dans la classification présentée ci-dessous.

a) **Le niveau de base** : Il est lié à la machine et au logiciel utilisés. Il constitue la partie non standard à modifier pour passer à un autre matériel. Il contient essentiellement :
   - les entrées/sorties de base liées aux médias utilisés,
   - les fonctions BASIC non standards,
   - les procédures de codage interne des caractères alpha-numériques et semi-graphiques.

   Le codage interne est nécessaire pour assurer une indépendance par rapport au matériel utilisé.

b) **Le niveau des objets de base** manipulés par les primitives : il s'agit des procédures qui manipulent :
   - les informations textuelles ou semi-graphiques,
   - les points d'ancrage de l'écran,
   - les zones écrans,
   - les messages sonores sur bande magnétique,
   - les messages visuels sur diapositives.

   Tous ces objets sont codés au niveau du noyau, pour assurer une indépendance entre le didacticiel et le matériel.

c) **Le niveau général de l'EAO** : Il regroupe les primitives MEFIA-EAO générales c'est-à-dire indépendantes du didacticiel. Ce sont essentiellement :
   - des filtres (opérations de mise sous forme canonique), utilisés fréquemment comme la banalisation majuscules / minuscules
   - certaines analyses de réponse courantes comme la comparaison terme à terme ou la recherche de mots-clés
   - les opérations d'inhibition ou d'activation de requêtes.

d) **Le niveau spécifique au didacticiel produit** : Il contient des primitives MEFIA-EAO dont la réalisation est spécifique au didacticiel comme :
   - la réception d'une réponse avec acceptation de la requête : la fonction de réception est générale à l'EAO, mais le protocole d'accès à une requête est lié au didacticiel et peut être un choix pédagogique,

- des primitives de présentation de scènes spécifiques comme la présentation de l'écran en début de dialogue,

- les schémas généraux prédéfinis qui correspondent à des algorithmes spécifiques aux didacticiels. Ces schémas sont définis par l'auteur en collaboration avec l'informaticien (cf §2.3), et apparaissent fréquemment dans la description des dialogues. Ils permettent de faciliter le travail de programmation à partir de la spécification des auteurs.

La bibliothèque est constituée au fur et à mesure de l'expérience acquise dans la description et la programmation de didacticiels. Les primitives sont définies en fonction des besoins par les auteurs, et correspondent, en général, à un symbole dans le dossier de description d'un dialogue (cf §2.3).

L'ensemble des primitives ainsi définies peut servir de base à la conception d'outils plus évolués destinés à des auteurs pour la spécification du didacticiel.

4.6.3 - Les problèmes de gestion de la bibliothèque

Un didacticiel n'est pas un logiciel figé. Il est sujet à de nombreuses modifications :

- Au moment du test par l'auteur : il y a toujours une distance entre la vision que l'auteur se fait du dialogue qu'il spécifie et sa réalisation ; ce phénomène est vérifié chaque fois qu'il y a collaboration entre des informaticiens et des utilisateurs de l'informatique. Cette distance diminue avec l'expérience acquise par l'auteur.

- A l'expérimentation, où l'on constate presque toujours certains effets non souhaités par les auteurs, sur les apprenants (réactions non prévues).

Dans les modifications apportées, on est amené à créer, supprimer ou modifier certaines primitives, essentiellement les schémas prédéfinis qui sont les plus proches du didacticiel.
L'introduction ou la suppression d'une primitive MEFIA-EAO de la bibliothèque engendre les opérations suivantes :

- La mise à jour des informations transmises à l'analyseur.

- La modification du graphe des appels des procédures ce qui implique de recalculer sa fermeture transitive.

- L'introduction ou la suppression du code BASIC de la primitive.

- La mise à jour des informations transmises au traducteur (les noms de objets BASIC par lesquels se fait la transmission des paramètres et des résultats).

D'autre part, on peut être amené à utiliser une nouvelle machine cible. Ceci implique de gérer une version supplémentaire de la bibliothèque. Celle-ci comprend :

. Un code BASIC adapté à la machine pour les procédures du noyau. La réalisation de certaines procédures peut être plus ou moins complexe suivant les possibilités offertes par la version de BASIC disponible. Dans certains cas, il est nécessaire d'écrire des parties de code en langage machine pour réaliser des fonctions absentes dans la version disponible.

. Un graphe des appels des procédures qui peut différer au niveau des appels entre des procédures du noyau.

Un ensemble d'outils qui permettent une mise à jour facile de ces informations, et en assurent la cohérence, a été réalisé. Il forme le sous-système de gestion de la bibliothèque.

4.6.4 - Les outils de gestion et de maintenance

Nous présentons ici les outils réalisés pour résoudre les problèmes exposés au paragraphe précédent. Les informations de la bibliothèque sont gérées par un sous-système qui comporte les composants suivants :

- Deux outils de **mise à jour** :
  - un premier outil qui permet la modification du code BASIC des procédures,
  - un second qui assure la mise à jour des informations associées.

- Un outil de **contrôle** qui vérifie que pour chaque procédure définie, le code BASIC associé existe ; et réciproquement, qu'à chaque procédure BASIC, les informations correspondantes existent bien.

- Un outil qui calcule la **fermeture transitive** d'un graphe donné.

- Un outil de **documentation** qui produit des documents relatifs aux procédures gérées.

Le choix d'assurer la mise à jour par deux outils séparés plutôt que par un seul, est lié à la nature des modifications faites par chacun d'eux. En effet, la correction du code BASIC correspond à une mise à jour de la **réalisation** de la procédure, alors que la modification des informations associées concerne en général sa **spécification**. D'autre part, seules les primitives MEFIA-EAO sont concernées par la mise à jour d'informations fournies à l'analyseur et au traducteur.
Dans certains cas, la modification profonde du code d'une procédure entraine une mise à jour des informations associées : le changement d'un nom d'objet BASIC pour la transmission d'un paramètre ou d'un résultat d'une primitive, l'ajout ou la suppression d'un appel de procédure ayant une influence sur le graphe des appels.

En général, le code BASIC est susceptible d'être modifié plus souvent que les informations associées. De plus, le fait de séparer la mise à jour en deux opérations, évite de créer un outil de taille trop importante qui risque de poser des problèmes d'implantation sur le micro-ordinateur de production. Le défaut de cette séparation est qu'elle est source d'erreurs de cohérence, si l'utilisation des deux outils n'est pas systématique lorsque c'est nécessaire. L'outil de contrôle assure un minimum de cohérence entre les objets gérés par les deux outils.

Spécification des composants du sous-système

a) L'outil de maintenance du code BASIC des procédures

Il permet pour une procédure donnée :
- l'**intégration** à la bibliothèque de son code BASIC,
- la **suppression** de code,
- l'**extraction**, sans suppression, de son code et son rangement dans un fichier séparé.

La modification du code d'une procédure consiste à supprimer la version périmée du code et à la remplacer par la nouvelle version.

données : Il s'agit d'un outil interactif. L'utilisateur fournit successivement :
- le nom de la machine dont on manipule la version des procédures,
- la liste des procédures dont le code est à supprimer,
- la liste des procédures dont le code est à intégrer à la bibliothèque,
- la liste des procédures dont on veut extraire le code et le récupérer dans un fichier de travail.

Autres données fournies au programme :
- le code BASIC des procédures à intégrer
- la bibliothèque concernée par les opérations de mise à jour.

résultats : - la bibliothèque modifiée,
- le code des procédures "extraites", rangé dans un fichier de travail.

Remarque : Le choix de procéder à la modification d'une procédure par la suppression de la version périmée et insertion de la nouvelle version, permet de réduire la complexité de réalisation des outils de mise à jour.

b) L'outil de maintenance des informations associées à chaque procédure

C'est l'outil qui assure la modification de la bibliothèque interne du traducteur, c'est-à-dire les noms prédéfinis dans la notation MEFIA-EAO. Il permet :
. l'intégration à la bibliothèque, de la spécification des nouvelles primitives et procédures anonymes,
. la suppression des informations concernant la spécification des primitives et procédures anonymes.

données : C'est un outil interactif ; l'utilisateur fournit successivement :
- la liste des procédures à supprimer,
- les informations concernant les procédures à intégrer, c'est-à-dire :
  . pour une primitive MEFIA-EAO (procédure "nommée") :
    - son nom MEFIA-EAO,
    - son type (action, modification, fonction, opérateur),
    - le nombre et le type de ses paramètres et de ses résultats,
    - les noms des objets BASIC correspondant à chaque paramètre et à chaque résultat,
    - le nombre et la liste des procédures appelées dans le code de la procédure.
  . pour une procédure auxilliaire ("non nommée") :
    - uniquement le nombre et la liste des procédures appelées dans le code BASIC.

résultats : Les informations fournies aux autres composants du système sont mises à jour :

- les informations fournies à l'analyseur (spécification des primitives),
- les informations fournies au traducteur pour optimiser les appels de primitives (noms des objets BASIC qui assurent la transmission des paramètres et des résultats),
- le graphe des appels des procédures de la bibliothèque,
- le "catalogue" des procédures contenues dans la bibliothèque (pour effectuer des contrôles).

c) L'outil de calcul de la fermeture transitive d'un graphe d'appels

Ce programme permet de recalculer, après des modifications effectuées par l'outil présenté précédemment, la fermeture transitive du graphe des appels, c'est-à-dire, pour chaque procédure contenue dans la bibliothèque, la liste complète des procédures nécessaires à son exécution.

Le résultat de ce calcul est transmis à l'éditeur de liens.

données : - graphe des appels.
- nom de la version de la bibliothèque concernée (machine cible).

résultats : - fermeture transitive du graphe d'appels.

d) L'outil de contrôle :

Il permet d'assurer un minimum de cohérence entre les opérations effectuées par les deux outils de mise à jour.

données : - "catalogue" des procédures de la bibliothèque
- code BASIC des procédures
- nom de la version de la bibliothèque concernée
- erreurs de cohérence rencontrées

résultats : Liste des procédures pour lesquelles il existe un code BASIC, et pas de spécification, et réciproquement, procédures pour lesquelles il existe une spécification et pas de code BASIC.

e) L'outil de documentation de la bibliothèque :

Il s'agit d'un ensemble de programmes simples qui fournissent à l'utilisateur du système, des informations relatives au contenu de la bibliothèque. Ils produisent, sous une forme externe lisible, les informations gérées :

- la spécification des primitives MEFIA-EAO enregistrées,

- les informations transmises au traducteur (pour optimisation des appels de primitives),

- le graphe des appels des procédures pour une version donnée de la bibliothèque,

- le code BASIC des procédures spécifiées par l'utilisateur, dans une version donnée.

4.6.5- Conclusion

L'utilisation d'une bibliothèque de procédures adaptées aux besoins, est nécessaire pour faciliter la production des didacticiels [MoM 83].
La structuration par couches des procédures, et les outils qui les gèrent, permettent une grande souplesse de travail, tant au niveau de la portabilité des procédures qu'à celui de la modification ou de la création de primitives MEFIA-EAO prédéfinies.

a) la portabilité :

L'adaptation des procédures à une autre machine implique simplement :

- la création d'une nouvelle version de la bibliothèque en adaptant le noyau.
- la création du graphe des appels, à partir du graphe existant ; il peut différer de celui d'une autre version, au niveau des appels entre des procédures du noyau.

Deux versions de la bibliothèque ont été réalisées au cours du projet : l'une adaptée au site de production, l'autre au site d'expérimentation.

b) La modification de procédures

Une modification sur la réalisation ou la spécification d'une primitive, se fait très facilement à l'aide des outils de mise à jour définis. La maintenance par l'utilisateur du système est facilitée par l'outil produisant des documents relatifs aux procédures gérées.

c) La création de nouvelles primitives prédéfinies

L'intégration d'une nouvelle primitive au système de production se fait simplement, comme pour les modifications, à l'aide des outils de mise à jour définis. Cette possibilité est beaucoup exploitée car elle permet de se définir les primitives au fur et à mesure des besoins. Elle permet de créer facilement de nouveaux schémas, adaptés aux besoins des auteurs. Ainsi, le niveau d'expression de la notion peut être élevé et adapté à la description du didacticiel.

4.7 Conclusion

Les outils constituant l'environnement de production correspondent à des fonctions "classiques" de développement de logiciel. Pour leur réalisation, des méthodes et des techniques de programmation et de génie logiciel ont été mises en oeuvre. La spécificité de l'environnement se situe :

- au niveau de la nature des objets manipulés, indépendante d'un matériel ou d'un logiciel spécifique,
- au niveau de la souplesse de travail notamment pour l'application de méthodes de programmation systématiques, pour la définition et l'intégration de nouvelles primitives, ce qui permet de "spécialiser" la notation aux problèmes de production de didacticiels.

Le traducteur ne permettant pas, dans son état actuel, de traduction séparée, un certain nombre de modules spécifiques, fréquemment utilisés dans la description des dialogues, sont prédéfinis.
Pour construire un autre didacticiel à partir de l'environnement réalisé, la démarche à entreprendre consiste dans un premier temps, à écrire des dialogues dans la notation en termes des primitives prédéfinies générales à l'EAO. Dans un deuxième temps, elle consiste, à partir de la première expérience acquise, à définir les schémas adaptés aux besoins des auteurs, les intégrer à la notation, et ainsi faciliter la description et la production du didacticiel.

Les problèmes d'implantation des programmes produits sur micro-ordinateurs (cf. §3.6.2) subsistent. Un didacticiel constitue un logiciel de taille importante, et des techniques de recouvrement sont à mettre en oeuvre au niveau du micro-ordinateur. Le chapitre suivant fait l'objet de la présentation des méthodes et des techniques appliquées pour résoudre ces problèmes.

CHAPITRE V

L'IMPLANTATION DES DIDACTICIELS PRODUITS SUR MICRO-ORDINATEURS

Au cours de la production des didacticiels, des problèmes d'implantation des programmes sur le site de diffusion ont été mis en évidence (cf. §3.6.2). Ces problèmes sont dûs aux ressources limitées des micro-ordinateurs, tant du point de vue du matériel que du point de vue du logiciel disponible. Il s'agit essentiellement :

- de la place mémoire "utilisateur" réduite,
- des possibilités de recouvrement des programmes, plus ou moins rudimentaires,
- des temps de chargement et d'exécution des programmes plus ou moins importants,
- de la capacité de stockage limitée des disquettes et des autres supports d'informations comme les cassettes et les paniers de diapositives.

5.1 - Résolution des problèmes de place mémoire

Pour résoudre les problèmes de place mémoire (cf. §3.6.2.1), il est nécessaire d'appliquer une stratégie de recouvrement des programmes. Celle-ci est liée :

- aux possibilités techniques de recouvrement offertes par le logiciel disponible,
- à la nature des programmes produits et à leurs relations de dépendance.

5.1.1 - Les possibilités de recouvrement disponibles sur micro-ordinateurs

Les possibilités de recouvrement de programmes varient d'une version de BASIC à une autre. Nous présentons ici les plus courantes, de la plus élémentaire à la plus évoluée.

Remarque1 : Nous ne prenons pas en compte ici le recouvrement non automatique, c'est-à-dire le chargement de programmes qui interrompt l'exécution en cours, et demande une intervention de l'utilisateur par une directive BASIC pour la poursuite de l'exécution. Il faut éviter d'exploiter une telle possibilité, car elle est source d'erreurs de l'utilisateur, ce qui diminue la fiabilité des programmes.

Remarque2 : Certains micro-ordinateurs puissants ont un système d'exploitation qui gère une mémoire "utilisateur" virtuelle ; c'est le cas de certains matériels équipés d'un disque dur et d'une mémoire centrale de taille importante. Nous ne nous intéressons pas ici à ces configurations de coût plus élevé, pour lesquelles l'implantation du didacticiel produit ne pose pas de problème.

Pour faire du recouvrement automatique, on utilise une instruction BASIC : "CHAIN", **disponible sur la majorité des interpréteurs et compilateurs** ; elle effectue des opérations différentes suivant la version utilisée. Nous avons ici un exemple typique d'instruction présentant une sémantique différente d'une version à une autre.

5.1.1.1 - Le "chaînage" de programmes indépendants :

L'instruction BASIC "CHAIN", permet de lancer l'exécution du **programme** suivant à exécuter ; elle a pour effet :

- d'effacer le contenu de la mémoire utilisateur, variables comprises,
- de fermer tous les fichiers qui étaient ouverts,
- de charger le programme spécifié en mémoire,
- de lancer son exécution.

Cette possibilité présente l'inconvénient important de perdre les valeurs de toutes les variables d'un chargement à un autre. La transmission d'informations entre deux programmes chargés consécutivement ne peut se faire qu'au moyen d'un fichier de données, ce qui ralentit très sensiblement l'exécution.
Cette forme de chaînage se rencontre essentiellement sur des interpréteurs BASIC.

5.1.1.2 - Le "chaînage" de segments de programmes

L'instruction BASIC "CHAIN", permet ici de lancer l'exécution du segment de programme suivant à exécuter ; elle a pour effet :
- d'effacer le segment de programme en mémoire,
- de charger le segment spécifié,
- de lancer son exécution.

Les valeurs des variables ne sont pas modifiées.
Cette possibilité permet d'exécuter des programmes séquentiels de taille importante, ou d'exécuter successivement des segments indépendants les uns des autres, mais qui ont des informations en commun.

5.1.1.3 - Le recouvrement de segments de programmes

Une instruction BASIC : "OVERLAY", permet de charger en mémoire le segment de programme spécifié. Après le chargement, le programme continue à s'exécuter en séquence, à la suite de l'instruction "OVERLAY". Aucune information n'est détruite, à l'exception des lignes de programme recouvertes par le segment chargé. Certaines versions de BASIC permettent de réaliser l'opération équivalente par d'autres instructions. Effet de l'instruction :

. en BASIC interprété : les lignes constituant le segment sont chargées en mémoire
  - soit à partir d'un numéro de ligne spécifié dans l'instruction, ce qui implique une renumérotation des lignes au cours du chargement,

- soit à partir du numéro de la première ligne du segment.

Les lignes du segment chargé sont fusionnées avec les lignes du programme en mémoire : chaque ligne du segment **remplace la ligne de même numéro** du programme en mémoire. Ceci implique que pour pouvoir se recouvrir, tous les segments doivent avoir leurs lignes numérotées de façon identique, au chargement. Dans certaines versions de BASIC [ADZ 83], on peut détruire par programme une séquence de lignes en mémoire. Dans ce cas, si avant le chargement d'un segment on détruit les lignes du segment précédent, il n'y a plus de lignes à interclasser, et donc plus besoin de numérotation identique des lignes.

. **en BASIC compilé** : les segments de programme sont compilés séparément ; les programmes objets sont en général en binaire absolu. Le chargement d'un segment de programme consiste à charger le code à l'adresse d'implantation spécifiée par le programmeur dans le programme source.

Cette dernière forme de recouvrement est la plus intéressante : elle permet, en cours d'exécution, de charger un segment de programme, en conservant le contrôle ; cela permet de gérer la mémoire, en chargeant les segments nécessaires à l'exécution du programme en cours. Elle est néanmoins plus rarement disponible. Seules les versions de BASIC assez complètes offrent cette facilité. De plus, le temps de chargement est parfois plus important que pour un chaînage, du fait que l'opération s'effectue ligne par ligne. On ne peut donc pas toujours exploiter cette technique, lorsque les délais de réponse de l'ordinateur doivent être courts.

Remarque : Sur les micro-ordinateurs à utilisation domestique, aucune possibilité de recouvrement n'est en général prévue. Le programmeur doit, soit faire du chaînage non automatique de programmes, soit programmer lui-même un mode de recouvrement, en langage machine.

En fonction des possibilités de recouvrement offertes par le logiciel de la machine cible, il faut définir une stratégie de recouvrement des programmes et de gestion de la mémoire, adaptée.
Au cours de l'exécution d'un dialogue, le temps de réponse doit être court, afin de maintenir l'attention de l'apprenant. Les performances du didacticiel sur le site de diffusion, sont en partie fonction de la stratégie définie.

5.1.2 - Stratégies de recouvrement adaptées

A chaque mode de recouvrement que nous venons de présenter, une organisation adaptée des programmmes est à définir. Les relations de dépendance entre les programmes du didacticiel sont exploitées pour définir cette organisation.

5.1.2.1 - "Chaînage" de programmes

Pour exploiter efficacement le "chaînage" des programmes, il faut faire du recouvrement entre des entités du didacticiel, qui partagent le moins d'informations possible. Ceci évite d'avoir un trop grand nombre d'informations à transmettre d'un programme à un autre, par l'intermédiaire de fichiers.

Le recouvrement au niveau des dialogues paraît le mieux adapté ; en effet chaque dialogue forme un tout et traite un point pédagogique précis (cf. §1.2).
Après un dialogue, le programme suivant à exécuter est l'unité de liaison qui lui est associée. Les informations à transmettre à l'unité de liaison sont celles de l'environnement dialogue qui servent :

- au calcul du dialogue suivant à exécuter,
- à la mise à jour de l'environnement didacticiel.

Des information sont globales à tout le didacticiel : l'environnement apprenant (cf. §1.1.2) et l'environnement didacticiel (cf. §1.1.3).

Ces informations sont contenues dans des fichiers. On peut choisir de ne les charger en mémoire qu'à l'initialisation des seuls dialogues qui les consultent ; cela permet de gagner souvent le temps de leur chargement, s'ils sont utilisés par un nombre faible de dialogues.

Suivant la place occupée en mémoire par un dialogue, on peut :

- soit charger l'unité de liaison associée, en même temps que le dialogue,
- soit ne la charger qu'à la fin de l'exécution, à la place du dialogue.

Le programme constituant un dialogue est formé :

- d'une procédure principale traduisant le graphe des situations pédagogiques qui le forment,
- des procédures correspondant aux situations pédagogiques et aux unités de décision,
- des primitives prédéfinies utilisées par les situations pédagogiques, et les procédures nécessaires à leur exécution.

Cette organisation du recouvrement au niveau des dialogues présente des problèmes :

- de nombreuses primitives de la bibliothèque sont utilisées dans tous les dialogues. Elles constituent une partie importante de ceux-ci, ce qui pose un problème d'efficacité : elles sont rechargées à chaque fois en mémoire avec le dialogue, ce qui augmente le temps de chargement,
- les dialogues ont parfois une taille trop importante pour tenir en mémoire, ce qui implique de faire du recouvrement au niveau des situations pédagogiques.

Le recouvrement au niveau des situations pédagogiques pose des problèmes d'efficacité importants lorsque le logiciel disponible n'offre que le "chaînage" de programmes comme moyen de recouvrement. Les informations à transmettre d'une situation pédagogique à une autre sont nombreuses ; il en résulte une augmentation sensible des temps de chargement, dûe à la transmission par un fichier de ces valeurs.

Un logiciel BASIC n'offrant que cette possibilité de recouvrement sera donc assez mal adapté à l'implantation de didacticiels. Seuls peuvent être utilisés facilement, les didacticiels formés de dialogues courts, donc simples. L'analyse de réponses ne peut dans ce cas qu'être peu développée, et de ce fait le didacticiel n'être que de qualité plutôt basse.

5.1.2.2 - "Chaînage" de segments de programme

Pour exploiter le "chaînage" de segments de programme de façon efficace, la même organisation de recouvrement, au niveau des dialogues, peut être mise en oeuvre. Les problèmes soulevés dans le cas précédent sont moins importants : si les dialogues ne tiennent pas en mémoire, le recouvrement au niveau des situations pédagogiques est possible, sans qu'il y ait inflation au niveau des temps de chargement. Les objets du programme n'étant pas définis au cours du chargement d'un segment, la transmission des informations entre les situations pédagogiques n'est plus coûteuse.

Pour les didacticiels constitués de dialogues de taille importante l'exécution d'un dialogue donne lieu au chargement

- de situations pédagogiques séparément,
- de l'unité de liaison associée après exécution de la dernière situation pédagogique.

Les informations globales à tout le didacticiel peuvent être chargées à l'initialisation et conservées en mémoire pendant toute la session, si elles n'occupent pas trop de place en mémoire.

Le chaînage de segments de programmes paraît beaucoup mieux adapté à l'implantation des didacticiels que le "chaînage" de programmes. Le problème du temps de chargement important subsiste : les primitives de la bibliothèque communes aux segments sont rechargées à chaque fois. Ce défaut pose le problème du maintien de l'attention de l'apprenant, pendant le chargement de la situation pédagogique suivante.

5.1.2.3 - Recouvrement de segments de programme

Par le **recouvrement de segments** de programme, il est possible de faire des chargements sans effacement de tout le programme résidant. Le problème des chargements multiples des procédures communes à plusieurs segments exécutés consécutivement peut être résolu : on conserve ces procédures en mémoire **pendant l'exécution de tous les segments qui les partagent.** Ainsi, les primitives utilisées dans un dialogue sont chargées à l'initialisation de celui-ci. Le recouvrement au niveau des situations pédagogiques peut se faire :

- soit par des **"chaînages" de segments**, possibilité qui est en général disponible en même temps que le recouvrement de segments,
- soit par **recouvrement de segments** ; dans ce cas, le programme principal du dialogue peut être résident, et effectuer le chargement des situations pédagogiques à l'exécution.

Si la partie résidante en mémoire, constituée par des procédures de la bibliothèque, prend trop de place, on peut effectuer du recouvrement entre certaines de ces procédures. Dans ce cas, les règles suivantes doivent être respectées pour assurer une exécution correcte du programme :

(1) Toute procédure n'appelant aucune autre procédure peut être recouverte et rechargée en mémoire au moment de son utilisation.
(2) Toute procédure chargée, ne peut recouvrir une procédure susceptible de l'appeler, directement ou indirectement.
(3) Toute procédure à charger, déjà présente en mémoire suite à un chargement précédent n'est pas à recharger.

Une procédure appelée par un nombre important de primitives, serait à recharger très fréquemment, et serait présente pratiquement en permanence en mémoire. Pour éviter de trop nombreux chargements, une telle procédure doit rester **residante** en mémoire, ainsi que les procédures qu'elle appelle.

Constitution de blocs séparés au niveau des procédures de la bibliothèque :

Toute primitive MEFIA-EAO appelle un ensemble de procédures. Cet ensemble peut constituer un bloc séparé, chargé au moment de l'appel de la primitive. De ce bloc sont à éliminer les procédures résidantes.
La constitution d'un bloc de procédures pour chaque primitive MEFIA-EAO, permet de gérer correctement la mémoire : par primitive exécutée, au maximum un chargement de bloc est effectué. Ceci peut être intéressant si le temps de chargement d'un bloc est très court ; sinon, on risque de perdre l'attention de l'apprenant, surtout pendant des opérations de calcul faisant intervenir plusieurs primitives.

Stratégie de recouvrement mise en place au niveau des didacticiels F.L.E. :

Le logiciel BASIC utilisé est un compilateur qui produit un code objet non translatable, chargeable à une adresse en mémoire spécifiée par le programmeur. Les possibilités de recouvrement disponibles sont :

- le chaînage des segments de programme,
- le recouvrement de segments de programme.

Les segments sont compilés séparément et chargés à l'adresse spécifiée dans le code BASIC.

Nous avons exploité les deux possibilités
- en faisant du chaînage entre les dialogues, ou entre les situations pédagogiques pour les dialogues de taille importante,
- en faisant du recouvrement au niveau de la bibliothèque, en constituant des blocs de procédures compilés séparément et structurés de façon à avoir au plus un chargement par exécution de primitive.

5.1.3 - Mise en place de la technique de recouvrement :

Nous présentons ici les méthodes et les techniques que nous avons mises en oeuvre pour mettre en place la stratégie de recouvrement définie ci-dessus.
Les dialogues ou les situations pédagogiques qui se recouvrent à l'exécution, sont traduits séparément en BASIC. Une interface entre ces programmes et les procédures de la bibliothèque structurées en blocs est nécessaire :

- pour assurer le chargement d'un bloc si nécessaire,
- pour effectuer le branchement au point d'entrée de la primitive appelée.

Le problème se situe au niveau de la définition des blocs de procédures à charger au cours de l'exécution. Le choix de faire un bloc par primitive MEFIA-EAO n'est pas acceptable, car sur notre site de diffusion, il augmente sensiblement le temps d'exécution.

5.1.3.1 - Méthode appliquée pour la définition des blocs

Définition de procédures résidantes :

Un ensemble des procédures résidantes a été defini à partir

- de l'inverse de la fermeture transitive du graphe des appels des procédures,
- de statistiques d'utilisation des primitives dans les dialogues.

Ont été définies comme résidantes, les primitives les plus utilisées dans les dialogues, ainsi que les procédures nécessaires à l'exécution d'un grand nombre de primitives.

Définition des blocs :

. Dans un premier temps, nous avons défini un bloc par primitive MEFIA-EAO non résidante, chaque bloc regroupant la primitive et les procédures non résidantes nécessaires à son exécution. La liste de ces procédures est donnée par la fermeture transitive du graphe des appels.

. Dans un deuxième temps, nous avons regroupé en un même bloc, les primitives qui partagent le même ensemble de procédures. Ce regroupement n'est bien entendu possible que dans la mesure où la taille du bloc obtenu ne dépasse pas celle de la zone réservée à son chargement.

5.1.3.2 - Mise en place sur le site de diffusion

La mémoire disponible a été décomposée en :
- une zone qui regroupe les informations manipulées par les dialogues et celles que manipulent les primitives,
- une zone réservée au recouvrement des dialogues, unités de liaisons et situations pédagogiques exécutées,
- une zone contenant les procédures résidantes de la bibliothèque,
- une zone réservée au recouvrement des blocs de procédures.

A priori, les blocs peuvent tous se recouvrir les uns les autres. Dans certains cas, l'exécution est ralentie, lorsque deux ou plusieurs blocs se recouvrent sans qu'il y ait communication avec l'apprenant, par exemple pendant une analyse de réponse.
Aussi, afin d'améliorer l'efficacité des programmes, la zone réservée au recouvrement de blocs est divisée en plusieurs parties. Les blocs susceptibles de se recouvrir successivement pendant un calcul, sont chargés dans des parties différentes. Ceci permet un gain important en temps de chargement.
Cette technique ne peut bien évidemment être mise en oeuvre que si la taille des blocs chargés en même temps ne dépasse pas la taille totale réservée au recouvrement des blocs.

L'interface entre les segments de programme du didactitiel et les procédures de la bibliothèque, doit effectuer à chaque appel de primitive :
- la recherche du bloc où se trouve la primitive appelée,
- le chargement de ce bloc en mémoire, s'il ne s'y trouve pas déjà,
- le branchement au point d'entrée de la primitive appelée.

Les informations nécessaires pour réaliser ces opérations sont :
- le nom (codé) de la primitive appelée (à fournir à l'appel),
- le nom du ou des blocs chargés en mémoire à un instant donné (géré à l'exécution),
- pour chaque primitive :
  - le nom du bloc où elle se trouve,
  - son point d'entrée (adresse absolue) ;

ceci implique la gestion de ces informations, et leur mise à jour à chaque modification des blocs. Les points d'entrée sont obtenus à la compilation des blocs.

Dans le programme exécutable obtenu après édition de liens, chaque primitive utilisée est remplacée par la séquence de code correspondant à son appel via l'interface ; c'est ainsi que l'éditeur de liens relie le programme à la bibliothèque structurée en blocs.
A la création d'un bloc, on procède de même pour les procédures résidantes appelées.

5.1.3.3 - Construction automatique des blocs

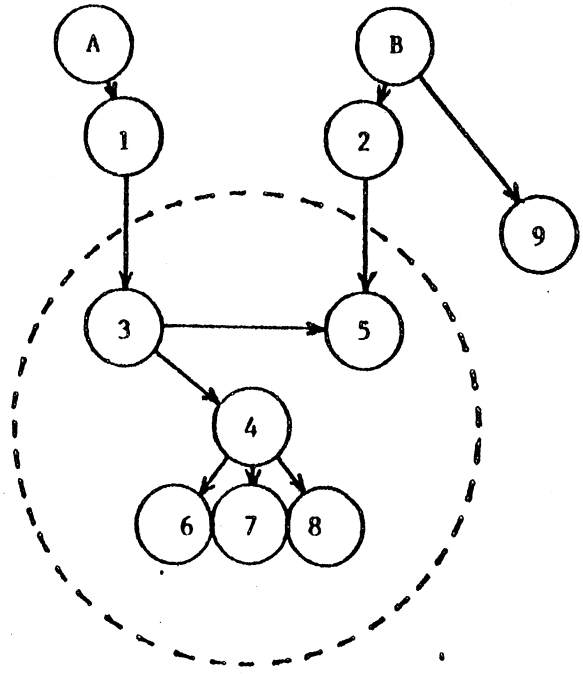
Les blocs ont été produits automatiquement à l'aide du même programme d'édition de liens : un précalcul de l'édition de liens entre les procédures de la bibliothèque est effectué.
Le bloc des procédures résidantes est produit automatiquement, à partir
- de la liste des primitives et procédures résidantes,
- du graphe des appels des procédures,
- de la bibliothèque.

Nous avons procédé de même pour la construction des autres blocs. Il a cependant fallu tenir compte des procédures résidantes nécessaires à l'exécution du bloc produit : celles-ci doivent être remplacées dans le bloc par la séquence d'appel à la primitive, via d'interface.

Pour obtenir le bloc après une édition de liens, il faut remplacer dans la bibliothèque, les procédures résidantes par leur séquence d'appel. Nous avons donc créé une seconde version de la bibliothèque destinée à la production automatique des blocs.

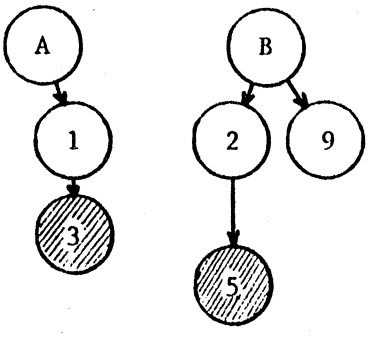
Le graphe des appels doit également être modifié : pour chaque procédure résidante, la liste des procédures appelées est remplacée par une liste vide. On calcule ensuite la fermeture transitive du graphe ainsi obtenu.

exemple : Soient A et B les primitives d'un bloc, et le graphe des appels de procédure suivant :



Si les procédures 1 et 2 ne sont pas résidantes et que la procédure 3 l'est, les procédures 4 à 8 le sont aussi (car nécessaires à l'exécution de 3).
Le bloc à produire doit contenir : A, B, 1, 2 et 9.
Les procédures 3 et 5 sont remplacées par la séquence de leur appel, via l'interface.

Le graphe modifié (avant fermeture transitive) est le suivant :



A chacun des noeuds A, B, 1, 2 et 9 correspond une primitive dans le bloc. Aux noeuds 3 et 5 correspondent les séquences d'appel aux primitives, via l'interface.

Le découpage de la bibliothèque en un bloc de procédures résidantes, et des blocs indépendants peut être automatisé par un algorithme plus ou moins complexe en fonction des données qu'il prend en compte. Celles-ci peuvent-être :

- le graphe des appels,
- la taille de chaque primitive,
- le nombre et la taille des zones de recouvrement et de la zone réservée aux procédures résidantes,
- la probabilité d'appel de chaque primitive,
- etc...

Les techniques de recouvrement que nous avons mises en oeuvre sont réapplicables pour toute machine disposant d'un compilateur BASIC, ou d'un interpréteur offrant un recouvrement de segments de programme efficace.

5.2 - Résolution des problèmes liés à la masse des informations et des programmes à gérer

Les didacticiels produits sont composés physiquement d'un ensemble de volumes aux possibilités de stockage limitées (disquettes, cassettes, paniers de diapositives). Ces volumes doivent être gérés à l'exécution, c'est à dire qu'il faut pouvoir avertir l'apprenant des changements de volume qu'il a à effectuer pour pouvoir poursuivre son travail. Pour celà, des fonctions qui facilitent la gestion des fichiers et des volumes sont nécessaires.

5.2.1 - Caractéristiques des fonctions de gestion de fichiers

Un ensemble de primitives a été défini pour faciliter la gestion des fichiers et des supports physiques :

- des primitives de manipulation de fichiers, qui assurent la gestion proprement dite (exemple : vérification de présence sur la disquette)
- des primitives de communication avec l'apprenant, pour lui demander d'intervenir en cas de changement de volume.

Les primitives sont utilisées principalement dans la description du programme principal (didacticiel) qui prend en charge cette gestion. Elles pourront apparaître dans la description d'un dialogue, si l'exécution de celui-ci nécessite des changements de volume.

5.2.1.1 - Les primitives de manipulation

Les primitives de manipulation permettent la gestion effective des volumes.

- Elles permettent de vérifier, avant le changement d'un segment de programme, que l'ensemble des volumes nécessaires à son exécution soient mis en place, et que les informations nécessaires y soient présentes ; cette dernière vérification n'est possible que sur les disquettes, les informations des autres volumes n'étant pas contrôlables par l'ordinateur.
- Elles assurent le chargement des fichiers en mémoire.

Ces primitives sont écrites en termes de fonctions de base du système d'exploitation de disquettes (s.e.d.) utilisé. Celles-ci sont en général accessibles au moyen d'**instructions BASIC non standard** (ouverture, lecture, écriture, fermeture, destruction de fichiers, etc...). Ces fonctions, sont toutefois présentes sur tous les systèmes d'exploitation de disquettes. Les procédures écrites en terme de ces fonctions font donc partie du noyau de la bibliothèque.

A titre d'exemples, nous donnons quelques primitives de manipulation utilisées fréquemment :

- vérification de la présence (ou l'absence) d'un fichier sur la disquette,
- vérification de la présence de la bonne disquette,
- vérification de la présence de la bonne cassette,
- destruction d'un fichier,
- création,
- ouverture,
- fermeture,
- chargement en mémoire du contenu d'un fichier.

5.2.1.2 - Les primitives de communication

En cas de changement de volume, ces primitives avertissent l'apprenant des manipulations à faire, par la présentation d'une scène ; l'exécution est interrompue. Après avoir effectué le changement de volume, l'apprenant indique la reprise du travail par la frappe d'une touche spécifique au clavier.

Les primitives de communication sont liées au didacticiel, car la scène présentée ne doit pas perturber la gestion de l'écran, et doit être homogène avec la forme de communication utilisée dans le reste du didacticiel. L'auteur peut définir la scène à présenter et la réponse à envoyer pour poursuivre l'exécution.

5.2.1.3 - Les informations manipulées

Afin de rendre les primitives indépendantes du s.e.d. disponible, des noms symboliques internes sont associés à chaque objet (volume ou fichier) utilisé. Une correspondance est faite entre le nom interne et le nom externe.
Les primitives gèrent un ensemble d'informations d'état qui mémorisent les noms des volumes utilisés à un instant donné.
De plus, pour chaque segment de programme à charger (dialogue, situation pédagogique, unité de liaison), les noms externes des objets nécessaires à son exécution sont regroupés dans un fichier consulté par les primitives.

Exemple : Pour un dialogue, le fichier géré par le SGF, contient l'ensemble des noms externes des objets nécessaires à son exécution :
- noms des volumes,
- nom du programme exécutable,
- nom du fichier de données,
- nom du fichier "trace" à produire,
- nom du fichier contenant les noms internes et externes des objets manipulés dans le dialogue, (utilisé à l'exécution du dialogue par les primitives de gestion de fichiers).

Tout objet est référencé par son nom interne à l'appel d'une primitive de gestion de fichier ; une fonction assure la correspondance entre le nom interne et le nom externe, par tabulation ou par calcul. Cette fonction est liée au s.e.d. et fait partie du noyau de la bibliothèque.

Afin d'éviter des erreurs dues à une production manuelle des informations du SGF, des outils simples ont été réalisés pour assurer leur production systématique.

5.2.2 - Organisation des informations sur les supports physiques

Dans les didacticiels F.L.E., les supports physiques, manipulables en même temps sont :

- deux disquettes magnétiques,
- une bande magnétique contenant les messages sonores
- un panier de diapositives (uniquement dans le didacticiel "passé-composé").

Nous n'avons pas utilisé plus d'un panier de diapositives ; il n'y a donc pas eu de problème de gestion de ce type d'information.

5.2.2.1 - Organisation des informations sur les disquettes

Deux disquettes sont disponibles en même temps. L'une correspond à la partie de **didacticiel** en cours d'exécution (disquette "didacticiel"), l'autre regroupe les informations liées à l'utilisateur du didacticiel et à son activité, c'est à dire **l'apprenant** (disquette "apprenant") ou **l'enseignant utilisateur** (disquette "enseignant").
Cette organisation permet de gérer plus facilement les modifications :

- les modifications concernant le contenu des dialogues affecteront les disquettes "didacticiels",
- les modifications concernant les informations liées à l'utilisateur affecteront les disquettes "apprenant" et "enseignant".

Les informations ont été organisées de façon à ne pas poser de problèmes lors du passage de disquettes 8" à des disquettes 5".

- **Informations stockées sur les disquettes "apprenant" :**

  . le système d'exploitation de disquettes (toujours figé),
  . le moniteur (programme principal) qui effectue le protocole d'entrée dans le système et fournit les modes de travail dans le didacticiel, prévus par l'auteur ; il est lié à l'utilisateur du didacticiel : on peut concevoir qu'un enseignant utilisateur ait d'autres modes de travail que l'apprenant.
  . les informations liées à l'apprenant et à son activité : environnements, traces.

- **Informations stockées sur les disquettes "didacticiels" :**

  . les dialogues ou segments de dialogues chargés successivement à l'exécution (programmes, données),
  . les unités de liaison,
  . les informations utilisées par les primitives de gestion de fichiers (liées au didacticiel),
  . les blocs de procédures chargés à l'exécution.

Dans les didacticiels FLE, des informations complémentaires sont accessibles au moyen des requêtes "lexique" et "verbe", qui donnent respectivement accès à un lexique français-arabe, et à la conjugaison d'un verbe donné.

Du point de vue informatique, il est préférable de stocker ces informations sur une disquette spécifique. Celà n'est pas toujours acceptable par les auteurs, si ces informations doivent être accessibles rapidement à l'apprenant.
Dans ce cas, on choisira de les stocker sur l'une ou l'autre des disquettes, ou les deux, suivant leur taille et la place qu'elles occupent sur une disquette.

5.2.2.2 - Organisation des informations sur les bandes magnétiques

L'accès aux informations d'une bande magnétique est séquentiel. Le magnétophone piloté par le micro-ordinateur dans le didacticiel "passé-composé" donne la possibilité d'accéder directement à un enregistrement donné.
Afin d'avoir un temps d'accès acceptable, les messages concernant un même dialogue doivent être regroupées.
Le problème se pose au niveau de la **modification** des messages : si la longueur d'un message augmente sensiblement, ou si le nombre de messages d'un dialogue est modifié, il faut reformater la bande. Un outil automatique, qui facilite la production de la modification des bandes magnétiques à l'aide de deux magnétophones a été étudié [Fal 82] et un prototype a été réalisé.

5.3 - Conclusion.

Les méthodes et les outils mis en oeuvre pour résoudre les problèmes d'implantation des didacticiels sur le site de diffusion ont permis une bonne exploitation des possibilités de la machine :

- par la définition d'une statégie de recouvrement des programmes adaptée,
- par la création de primitives de gestion de fichiers indépendantes du système d'exploitation de disquettes disponible.

Des problèmes subsistent néanmoins :

a - Certaines séquences de calcul, faisant intervenir des blocs de procédures de taille importante se recouvrant, sont assez lentes ; ceci limite les possibilités du didacticiel.

b - La gestion de la mémoire utilisateur est entièrement à la charge des informaticiens : ils choisissent les adresses d'implantations des procédures de la bibliothèque. Des adresses sont à mettre à jour après chaque modification de bloc. Un outil permettant l'automatisation de ces mises à jour faciliterait la maintenance des blocs. L'utilisation d'un compilateur BASIC produisant du code translatable et permettant la compilation séparée, permettrait une gestion plus souple de la mémoire.

c - Dans le cas de mémorisation de traces de taille importante, les possibilités de stockage des disquettes (5" double face, simple densité) nous paraissent trop faibles.

Nous constatons que, pour le type d'EAO que nous voulions réaliser, le matériel de diffusion expérimenté est d'une trop faible puissance. De nombreux outils construits n'auraient pas été nécessaires pour implanter les didacticiels sur des matériels plus puissants, tels que des micro-ordinateurs 16 bits munis de systèmes d'exploitation plus évolués.

D'autre part, nous n'avons pas exploité de possibilités graphiques dans les didacticiels, ce qui est pourtant important dans la communication avec l'apprenant [Lem 83]. L'utilisation du graphique demande en général des configurations de matériel plus puissantes, notamment au niveau de la place mémoire disponible.

L'évolution des matériels laisse supposer que les problèmes d'implantation évoqués et résolus ici, ne se posent plus sur les nouvelles générations de micro-ordinateurs. Toutefois, les techniques que nous avons développées permettent une exploitation intéressante de nombreux matériels de grande diffusion.

CONCLUSION

Vers un atelier de production de didacticiels

La production d'un didacticiel à l'aide de l'environnement que nous avons réalisé, se résume aux étapes suivantes :

- les pédagogues spécifient les dialogues au moyen d'un dossier normalisé rédigé dans un formalisme conçu par eux mêmes et très proche du français courant.

- les informations contenues dans ce dossier sont traduites manuellement et de façon systématique en un programme MEFIA-EAO, par les informaticiens.

- le programme est traduit automatiquement en langage BASIC standard.

- une édition de liens a lieu entre ce programme objet et la bibliothèque dans la version de la machine de production.

- les pédagogues et informaticiens font des tests et des mises au point sur le site de production.

- par une nouvelle édition de liens, on adapte le programme objet au site de diffusion où des tests avec les apprenants concernés peuvent être faits.

La démarche que nous avons suivie est basée sur le maintien de la spécificité des intervenants : nous avons voulu libérer les auteurs des contraintes informatiques afin qu'ils puissent se consacrer entièrement au travail pédagogique. Nous ne voulions pas proposer des outils aux auteurs, avant qu'ils n'aient bien défini leurs besoins.

D'autre part, le passage au programme informatique se fait **indépendamment d'un matériel ou d'un langage de programmation :**

La programmation se fait en appliquant des règles de description de didacticiels dans une notation algorithmique. Celle ci est enrichie de fonctions spécifiques aux didacticiels, qui facilitent leur description.
Des outils sont créés pour faciliter la production :
- ils sont conçus pour favoriser le transport des programmes produits sur des sites différents,
- ils assurent la fiabilité des programmes produits, et la facilité de leur modification, ce qui est nécessaire pour des produits qui ne sont figés qu'après une longue période de tests.

Les suites à MOSAIQUE

1) Réalisation d'outils proches de l'auteur

La mise en oeuvre des méthodes et des outils définis, pour la production de didacticiels de taille importante, a donné les éléments pour étudier des outils de production plus évolués destinés à des auteurs.

Deux études ont été effectuées :
. une première sur la spécification et le test de l'analyse de réponses, par la programmation logique [Luc 83],
. une seconde sur la spécification et l'utilisation d'objets graphiques pour la communication [Lem 83].

Les outils définis sont adaptés au type d'EAO appliqué dans le didacticiel FLE. Pour ces deux études, la même démarche de travail que pour le projet MOSAIQUE a été appliquée :
- modélisation des objets à spécifier,
- étude des problèmes de spécification de ces objets par les auteurs,
- réalisation d'une maquette qui permette aux auteurs de faire les spécifications dans un formalisme très proche du leur.

La réalisation est faite en utilisant les fonctionnalités générales issues des techniques mises en oeuvre, indépendamment d'une implantation particulière.

a) L'approche "programmation logique"

Cette étude a permis de définir des règles de description des dialogues, notamment des analyses de réponses, dans un langage de programmation logique. Ces règles sont très proches de la spécification des auteurs, et sont indépendantes d'une implantation particulière d'un langage de programmation logique.

Les analyses de réponses étudiées sont adaptées à l'enseignement des langues, et peuvent être facilement spécifiées par un auteur linguiste habitué au formalisme des grammaires.

b) L'approche "graphique"

Cette étude a permis d'introduire, dans le contexte MOSAIQUE, la manipulation d'objets graphiques dans les didacticiels. Un certain nombre de primitives de manipulation sont définies, indépendamment d'un logiciel graphique particulier, et en n'utilisant que des fonctionnalités standards. Suivant le logiciel graphique disponible, les fonctions évoluées à mettre en oeuvre sont directement utilisées, ou à programmer à partir des fonctions graphiques de base.

La maquette d'un éditeur d'objets graphiques a été réalisée. Elle permet à des auteurs la création et la modification des objets manipulés.

2) Réalisation d'autres didacticiels

A l'aide de l'environnement de production existant, la réalisation de didacticiels de programmation est actuellement en cours. Sur le plan pédagogique, ils privilégient une approche "lecture".

Cette réalisation permet la définition de nouvelles fonctions adaptées à ces didacticiels, qui complètent l'ensemble déjà existant [Mon 84] [Zam 84].

D'autre part, la bibliothèque de primitives spécialisées a été réécrite en PASCAL, en respectant la structure par niveaux des primitives ; ceci permet d'utiliser PASCAL comme support des didacticiels, pour les implantater sur des sites disposant de ce langage.

3) Adaptation au système DIANE

Le système DIANE est né au sein de l'Agence de l'Informatique [ADI 83] ; il est devenu la **norme nationale** en matière d'enseignement assisté par ordinateur.

Les composants des didacticiels produits sous DIANE sont créés et modifiés à partir d'éditeurs spécialisés destinés à des auteurs. Ils sont représentés en machine sous une forme interne normalisée, interprétable sur les sites de production et de diffusion. Cette forme interne a été conçue pour recevoir de nouveaux composants, ainsi que les éditeurs et les interpréteurs associés.

La structuration des didacticiels spécifiés sous DIANE est la même que celle que nous avons adoptée pour les didacticiels F.L.E. (chapitre 1). Les didacticiels que nous produisons doivent donc être pouvoir être facilement produits à partir des éditeurs de DIANE, et ainsi être normalisés. Les fonctions que nous avons définies pour leur réalisation pourront ainsi enrichir les fonctionnalités de DIANE déjà existantes.

D'autre part, les outils spécialisés pour la spécification des objets graphiques et des analyses de réponses doivent pouvoir être transformés pour produire de la forme interne DIANE, éventuellement par la création de nouveaux composants, et des interpréteurs associés.

L'expérience acquise au cours du projet nous a apporté les éléments nécessaires à la conception d'éditeurs proches des besoins et des formes de travail des auteurs.

4) Evaluation

Tout notre travail va servir de base à une étude de méthodes et d'outils pour l'évaluation des didacticiels et des systèmes de production, problème qui a été peu étudié jusqu'à présent en tant que tel.

REFERENCES BIBLIOGRAPHIQUES

[AbZ 83] S. ABOU-JAOUDE, T. ZEMB
Le Sbasic, un dérivé performant d'APL
L'enseignement, l'ordinateur et APL
Journées d'étude AFCET-APL. Avril 1983

[ACD 81] J.M. ADAM, Y. COSTE, H. DELORI, M.P. FLEURY
Implantation et utilisation de MEFIA sur micro-ordinateur.
Rapport intermédiaire du projet MOSAIQUE.
INPG - Laboratoire IMAG. Grenoble, Octobre 1981

[ADE 82] ADELE : un atelier de développement de logiciel (trois articles)
RR no. 299 - Laboratoire IMAG - Grenoble 1982

[ADI 81] Agence de l'Informatique (A.D.I.).
Rapport E.A.O. - Paris 1981

[ADI 83] Agence de l'informatique.
Système DIANE, Enseignement assisté par ordinateur.
Service formation - Paris Février 1983

[Adm 79] J.M. ADAM
Etude comparée des dialectes du langage BASIC. Définition d'un noyau.
Rapport interne - INPG - Grenoble, 1979

[Adm 80] J.M. ADAM
Construction méthodique de programmes : étude d'un environnement d'aide à la programmation systématique sur micro-système.
Rapport de DEA - INPG - Grenoble, Juin 1980

[Adm 81] J.M. ADAM
Règles de description d'objets pédagogiques en MEFIA
Rapport intermédiaire du projet MOSAIQUE.
INPG - Laboratoire IMAG. Grenoble, Octobre 1981

Bibliographie

[Avn 79] R.A. AVNER
Production of Computer-Based Instructionnal Materials in [One 79]

[BeF 82] H. BESTHOUGEFF, J.P. FARGETTE
Enseignement et Ordinateur
Cedic Fernand Nathan 1982

[Bos 82] G. BOSSUET
L'ordinateur à l'école.
P.U.F. L'éducateur - Avril 1982

[Cas 79] C. CASERY
Construction méthodique de programmes : réalisation d'un traducteur de la notation MEFIA vers PL/1. Généralisation à d'autres langages.
Rapport de DEA - INPG - Grenoble, 1979

[CCE 77] Cours de la Commission des Communautés Européennes
Les fondements de la programmation - Toulouse 1977
Edité par l'IRIA

[CDC 76] CONTROL DATA CORPORATION
Control Data PLATO system overview
Minneapolis, Minnesota 1976

[CDF 81] CONTROL DATA FRANCE
PLATO, système informatique spécialisé pour la formation.
Paris 1981

[CGS 78] P.Y. CUNIN, M. GRIFFITHS, P.C. SCHOLL
Aspects fondamentaux du langage MEFIA
Journées d'études sur la fiabilité des programmes dans les applications industrielles (IRIA-EDF Chapitre français de l'ACM) IRIA/EDF - Clamart, avril 1978

[CGV 80] P.Y. CUNIN, M. GRIFFITHS, J. VOIRON
Comprendre la compilation.
Spinger-Verlag 1980

[ChS 80] J. A. CHAMBERS J. W. SPRECHER
Computer Assisted Instruction : Current Trends and Critical Issues.
CACM Volume no. 6 June 1980

[CSV 76] P.Y. CUNIN, M. SIMONET, J. VOIRON
Méthodologie d'écriture de compilateurs. Une expérience du langage ALGOL 68
Thèse de docteur ingénieur - USMG, INPG Grenoble 1976

[CTI 81] CENTRE TECHNIQUE INFORMATIQUE
Etude micro-ordinateurs.
Ministère de l'Industrie - Paris, Mai 1981

[Cun 79] P.Y. CUNIN
Fiabilité et sécurité des programmes : propositions autour d'un langage d'essai.
Thèse d'Etat - CRIN, INPL Nancy, 1979

[Dav 83] J. P. DAVID
Plan de présentation d'un langage d'auteurs : AEIOU
C.I.A.P. Grenoble mars 1983

[DeG 81] M. DELAUNAY J.F. GRABOWIECKY
Note technique d'utilisation du transformateur de grammaires LL(1).
RR. no. 234. Laboratoire IMAG - Grenoble 1981

[Dub 78] J.P. DUBOURREAU
Construction méthodique de programmes : étude du langage MEFIA et d'un traducteur.
Rapport de DEA - INPG - Grenoble, 1978

[Faf 76] G. FAFIOTTE
Système d'exploration et de guidage didactique dans une base d'activités.
Rapport de recherche, laboratoire IMAG.

[Faf 78] G. FAFIOTTE
Projet Magister-Erasme. Laboratoire IMAG
Ecole Pluridisciplinaire IRIA mars 1978

[Faf 81] G. FAFIOTTE
Matériels pour l'E.A.O.
Journée de synthèse "utilisation des ordinateurs dans l'enseignement". Congrès AFCET. Novembre 1981

[FaI 82] G. FAFIOTTE, K. ITOH
Système de production de cassettes
Rapport intermédiaire du projet MOSAIQUE
INPG - Laboratoire IMAG - Grenoble Septembre 1982

[FFG 78] G. FAFIOTTE, J. FOUCHIER, J. GAUCHE, S. PAINVIN
CHOCS : Simulation sur ordinateur d'un phénomène physique destinée à une expérimentation physique.
Journal EPI 17. 1978

[FPS 76] G. FAFIOTTE, S.PAINVIN, P.C. SCHOLL
Insertion de l'Enseignement Assisté par Ordinateur dans l'Enseignement Secondaire, formation à l'EAO :
Analyse et propositions
RR. no. 30 Laboratoire IMAG - Grenoble février 1976

[Gal 77] M. GALINIER
A software design methodology and associated tools
dans [CCE 77]

[Gle 81] G. T. GLEASON
Micro-computer in education : the state of the art
Educational Technology March 1981

[GoH 74] G. GOOS,J. HARTMANIS (editors)
Compiler construction - an advanced course
Lecture notes in computer sciences, no. 21
Springer Verlag - Mars 1974

[Kea 82] G. KEARSLEY
Authoring Systems in Computer Based Education.
Communications of the ACM Volume 25 N-7 JULY 1982

[Lan 83] P. LANDRY
L'apport de l'ordinateur dans l'Enseignement, panorama des systèmes d'EAO en France.
L'Enseignement, l'Ordinateur et APL.
Journées d'étude AFCET - APL. Avril 1983

[Leb 80] P. LE BEUX
Introduction au Basic
Sybex 1980

[Lem 83] S. LEMOINE
Graphique et Enseignement Assisté par Ordinateur.
Rapport de DEA - INPG - Grenoble, Juin 1983

[LSV 77] M. LUCAS, P.C. SCHOLL, J. VOIRON
Apprentissage et utilisation du traitement séquentiel pour la construction de programmes
RR no. 74 - Laboratoire IMAG - Grenoble 1977

[Log 79] R.S. LOGAN
A State-of-the-Art Assessment of Instructional Systems Development in [One 79]

[Luc 83] A. LUCCI
Programmation logique et Enseignement Assisté par Ordinateur.
Rapport de DEA - INPG - Grenoble, Juin 1983

[Mar 83] J.F. Le MARECHAL
Un exemple d'analyse de réponse complexe d'un élève et sa modélisation en APL.
L'Enseignement, l'Ordinateur et APL.
Journées d'étude AFCET - APL. Avril 1983

[MIN 81] MINISTERE DE L'EDUCATION
Les technologies de communication au service de l'éducation. Quelques textes pour quelques réflexions actives. Ministère de l'éducation - Mars 1981

[Moi 84] H. MOITA MONTE
Thèse docteur-ingénieur à paraître
INPG - Grenoble 1984

[MoM 83] J.L. MOISY, H. MOISY-PAUGAM
Les fonctionnalités minimales d'un système EAO.
L'Enseignement, l'Ordinateur et APL.
Journées d'étude AFCET - APL. Avril 1983

[Mor 79] P. MORAT
Construction méthodique de programmes : réalisation d'un traducteur de la notation MEFIA.
Etude de quelques primitives de contrôle.
Rapport de DEA - INPG - Grenoble, 1979

[Mor 83] P. MORAT
Une étude sur la base de la programmation algorithmique : notation et environnement de travail.
Thèse 3ème cycle - INPG Grenoble 1983

[MOS 82] Projet MOSAIQUE : un didacticiel pour l'enseignement du français - langue étrangère.
Document pédagogique et manuel de l'utilisateur.
INPG - Laboratoire IMAG / ULLG - CUEF, Grenoble Jul. 1982

[MOS 83] Projet MOSAIQUE : création et mise en oeuvre de méthodes et outils informatiques d'aide à l'enseignement. Expérimentation sur un projet d'enseignement du français langue étrangère (rapport final).
INPG - Laboratoire IMAG / ULLG - CUEF, Grenoble Oct. 1983

[MRR 82] J. MOSSIERE, J. RAYMOND, Y. ROUZAUD
Representation interne et manipulation des programmes dans l'atelier de logiciel Adèle dans [ADE 82]

[One 79] H.F. O'NEIL Jr (editor)
Issues in Instructional Systems Development
Academic Press, Inc. 1979

[OnO 79] A.F. O'NEAL, H.L. O'NEAL
Author Management Systems in [One 79]

[Pai 81] S. PAINVIN
Cycle de préparation au projet
Rapport intermédiaire du projet MOSAIQUE
INPG - Laboratoire IMAG / ULLG - CUEF Grenoble Mars 1981

[Par 72] D.L. PARNAS
A technique for the specification of software modules with examples
CACM vol. 15, no. 5, pp. 330-336
May 1972

[PeG 68] M. PELTIER, M. GRIFFITHS
"Grammar Transformation as an aid to Compiler Production"
Centre Scientifique IBM Grenoble.
Etude FF2.0057.0 Mai 1968

[RoR 81] N. RODRIGUEZ J.F. REY
ARLEQUIN - Document de travail
Ministère de l'Education Nationale - Direction des lycées / INRP Paris octobre 1981

[SAP 81] M. SCHOLL, D. ABRY, S. PAINVIN
Un didacticiel pour l'enseignement du français - langue étrangère. Rapport de spécification pédagogique
INPG - Laboratoire IMAG Grenoble, Octobre 1981

[ScF 80] P.C. SCHOLL, G. FAFIOTTE
Création et mise en oeuvre de méthodes et d'outils d'aide informatique à l'enseignement
INPG - USMG Grenoble, Novembre 1980

[Sch 77.a] P.C. SCHOLL
Méthodologie de la programmation : une étude de cas : construction d'un index
dans [LSV 77] Grenoble 1977

[Sch 77.b] P.C. SCHOLL
Introduction à la récursivité et aux arbres
Support de cours - Institut de Programmation
USMG - Grenoble 1977

[Sch 78] P.C. SCHOLL
Le traitement séquentiel : une classe de problèmes et une méthode de construction de programmes
congrès AFCET-TTI Gif sur Yvette - Novembre 1978

[Sch 79] P.C. SCHOLL
Vers une programmation systématique : étude de quelques méthodes, techniques et outils.
Thèse d'état - Grenoble, Juin 1979

[Sch 80] Ma. SCHOLL
Soutien en français - langue étrangère, conçu et contrôlé à l'aide de micro-ordinateurs.
INPG - ULLG Grenoble, Novembre 1980

[Sch 83] Ma. SCHOLL
Etude de méthodes en enseignement assisté par ordinateur du français langue étrangère :
Résultats d'une expérience de production de didacticiels.
Rapport de DEA - IMSS/CUEF Grenoble, Octobre 1983

[SCM 80] P.C. SCHOLL, C. CASERY, P. MORAT
Introduction à une notation algorithmique.
USMG - INPG - Grenoble, Mai 1980

[ScS 72] P.C. SCHOLL, Ma. SCHOLL
Une expérience d'enseignement assisté par ordinateur du français. "Les techniques de l'informatique"
Congrès AFCET - Grenoble 1972

[SGC 78] P.C. SCHOLL, M. GRIFFITHS, P.Y. CUNIN
Construction méthodique et vérification systématique de programmes : éléments d'un langage
Congrès AFCET-TTI Gif sur Yvette - Novembre 1978

[Sim 80] J. C. SIMON
L'éducation et l'informatisation de la société
Rapport au Président de la République
La Documentation Française, Paris 1980

[TMS 79] Système TMS 115 Manuel de l'utilisateur
volume 4 : sous-système EAO (langage LOVE)
IUT LYON 1 / IRPEACS (CNRS) Lyon, Février 1979

[Val 73] C. VALE
Enseignement Assisté par Ordinateur : une expérience d'écriture de cours
R.A.I.R.O. p. 129 - 152 Janvier 1973

[Van 81] C. VAN DER MAST
A portable program to present courseware on microcomputers. Report of the Department of Mathematics and Informatics no. 81-07 Delft 1981

[Wir 71.a] N. WIRTH
The programming language PASCAL
Acta Informatica, vol. 1n No. 1, pp. 35-63

[Wir 71.b] N. WIRTH
Program development by stepwise refinement
Communications of the ACM, vol. no. 4, pp. 221-27 - April 1971

[Zam 84] J. ZAMBRANO
Thèse 3ème cycle à paraître
INPG - Grenoble 1984

[Zem 83] T. ZEMB
APL, Outil de modélisation de didacticiels
L'ordinateur, l'enseignement et APL
Journées d'études AFCET-APL Avril 1983

ANNEXE

Un exemple de session de travail avec le didacticiel FLE version 2

Après avoir mis en marche le micro-ordinateur,
mis les disquettes en place,
et procédé à l'initialisation du système,

- l'apprenant tape la commande MOSAIQUE,

- puis son nom.

Un bandeau de présentation du didacticiel apparaît.

Un didacticiel de Français - langue étrangère

pour arabophones

PRESENT - PASSE - FUTUR

Produit réalisé par l'IMAG,
avec la collaboration du CUEF

La liste des commandes disponibles s'affiche alors.

- l'apprenant choisit, par exemple,

de faire des exercices sur le FUTUR.

```
                Liste des commandes disponibles
                -------------------------------


0    arr@t


1    passé composé
2    présent
3    futur

4    verbe
5    mémento


Donnez le numéro de la commande choisie : 3
```

La liste des exercices du FUTUR apparaît.

- l'apprenant choisit, par exemple,

de faire l'exercice intitulé : les promesses.

La disquette contenant l'exercice demandé n'est pas en place.
Le système lui indique le numéro de la disquette à monter.

La partie basse de l'écran est réservée à ce type d'informations, ainsi qu'aux commentaires sur les réponses de l'apprenant et aux contenus de certaines requêtes.

```
                Liste des exercices du FUTUR          -->
                ----------------------------

0  arrêt

1  axe du temps                    11   puzzle 3
2  si la pollution continue        12   futurs irréguliers
3  puzzle 1                        13   que sera l'avenir ?
4  formation du futur              14   puzzle 4
5  les terminaisons du futur       15   bulletin de la météo
6  les promesses                   16   proverbes
7  puzzle 2                        17   puzzle 5
8  présent ? passé ? futur ?       18
9  les vacances                    19
10

Donnez le numéro de l'exercice choisi : 6
```

```
        Montez la disquette 1 sur l'unité 1
```

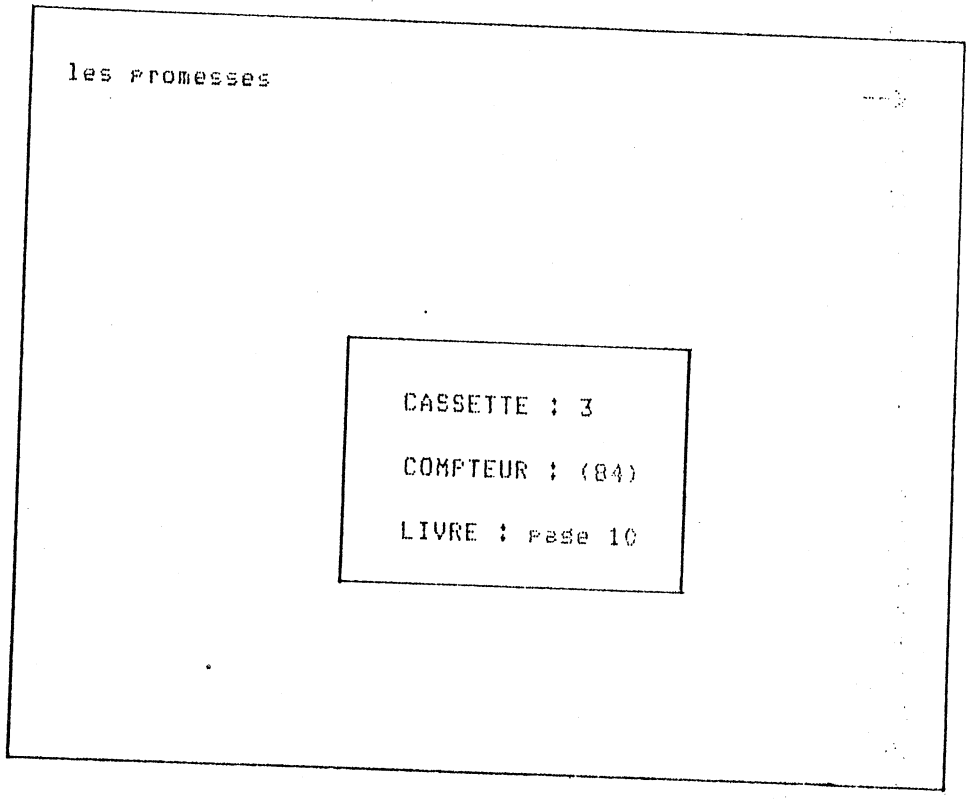
La flèche (en haut et à droite de l'écran) indique une pause du programme.

- quand l'apprenant est prêt, il l'indique au système à l'aide de la touche "fin de ligne".

L'exercice commence.

- l'apprenant met en place la cassette demandée,

- et ouvre le livre à la page indiquée.



- l'apprenant lit le texte de l'exercice,

- et regarde la page 10.

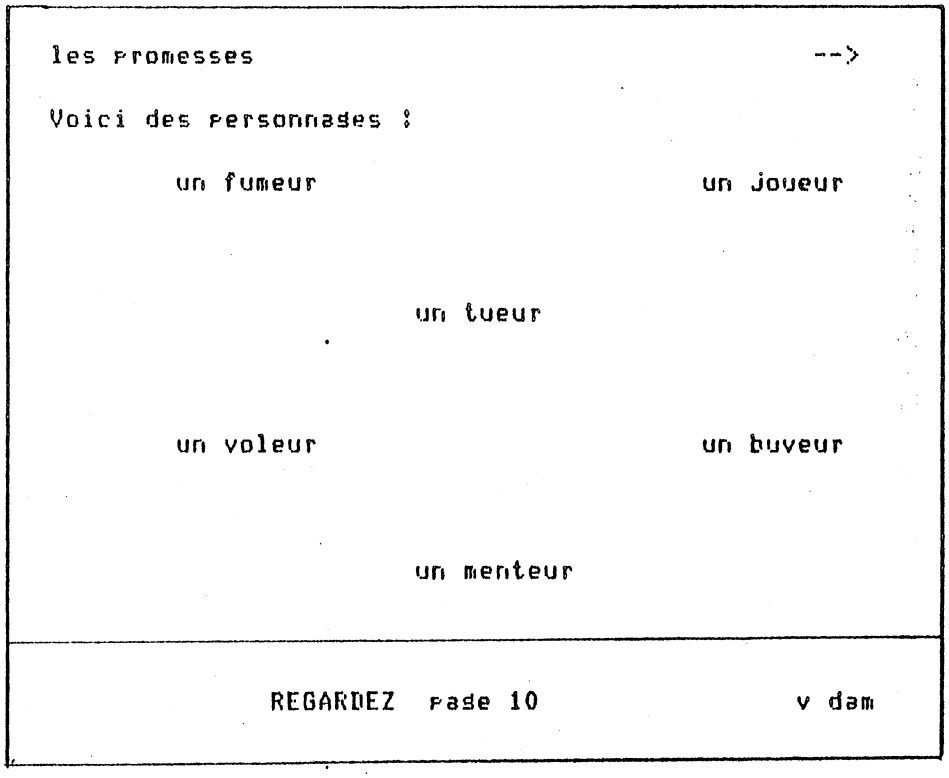
Les lettres (en bas et à droite de l'écran) correspondent à l'initiale des requêtes à sa disposition :

v pour consulter la conjugaison d'un verbe

d pour retourner au tout début de l'exercice

a pour arrêter l'exercice

m pour obtenir une aide technique



L'exercice consiste à relier les mots affichés et les dessins du livre.

- l'apprenant tape la lettre correspondante,

- et peut, à son 2ème essai, obtenir la réponse souhaitée en faisant appel à la requête R (réponse).

les promesses

Reliez les dessins (a,b,c,d,e,f) du livre aux personnages

un fumeur

d

un joueur

c

un tueur

a

un voleur

b

un buveur

—

un menteur

f

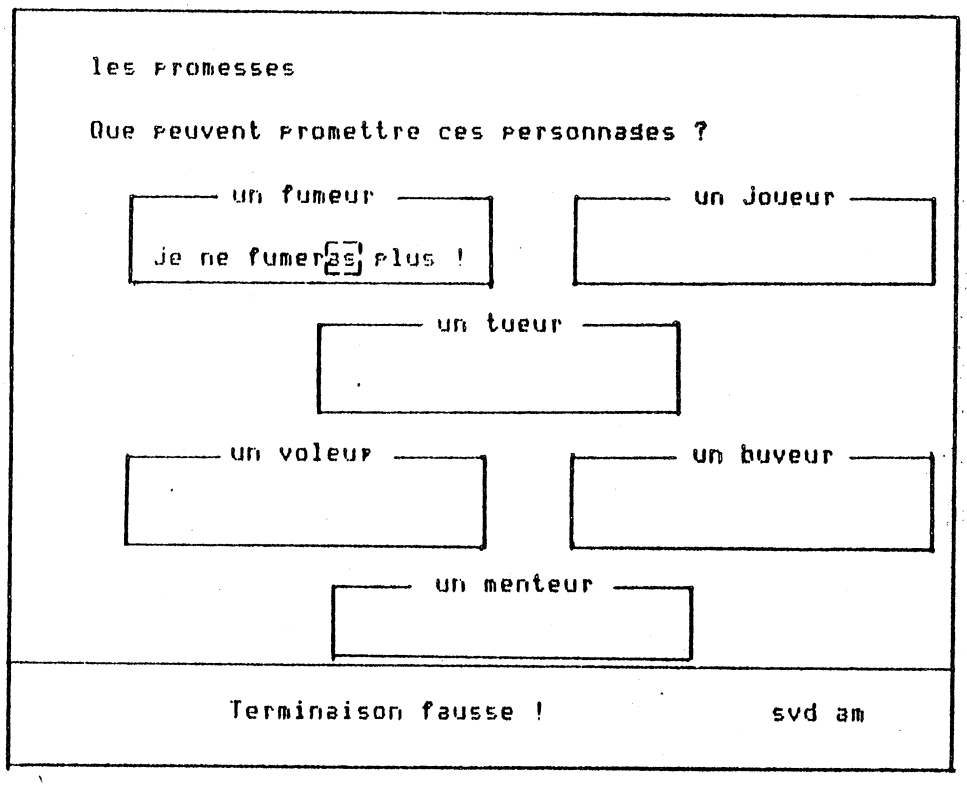
vrdam

L'exercice se poursuit par un autre travail. La consigne est accessible, en arabe, sur la cassette.

- l'apprenant tape le verbe qui convient, en le conjuguant.

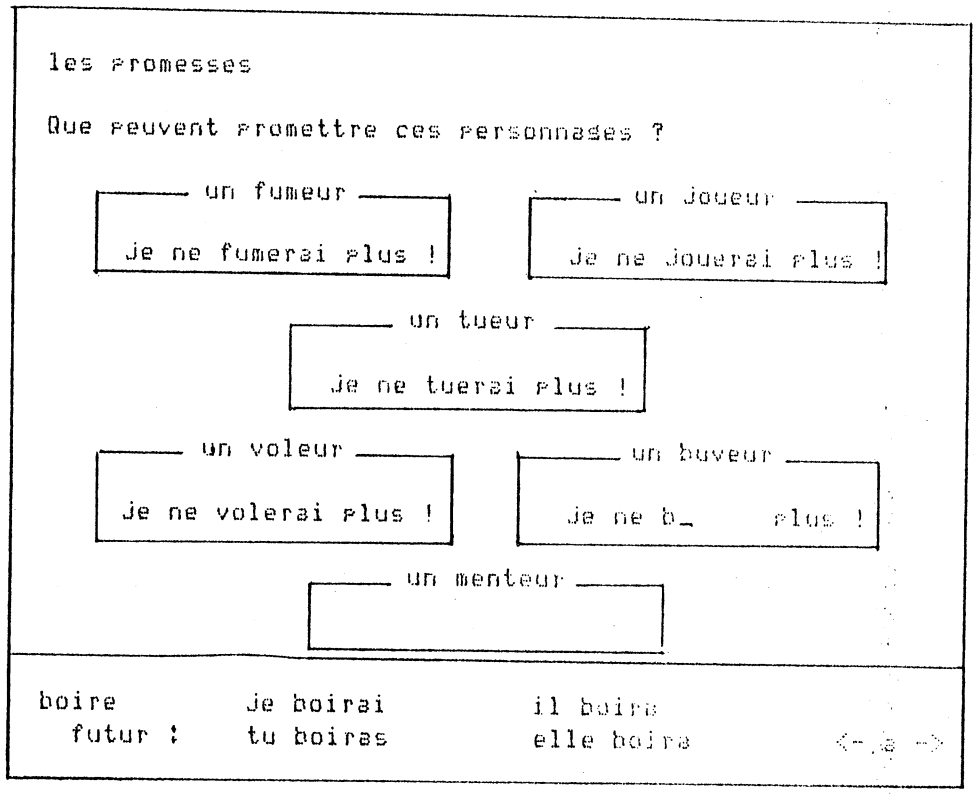
Le système vérifie la pertinence du choix du verbe,
détecte les fautes d'orthographe,
et indique l'emplacement de l'erreur (par clignotement) ainsi que la nature de celle-ci (dans le bas de l'écran).

- l'apprenant corrige sa réponse à l'aide d'un petit éditeur.



En cas de difficulté de conjugaison,

- l'apprenant peut appeler la requête V (verbe) et, par feuilletage, à l'aide des touches : <- pour aller en arrière,
  -> pour aller en avant,
  a pour arrêter,
  obtenir la conjugaison du verbe proposé, aux 3 temps.



En cas de difficulté de compréhension,

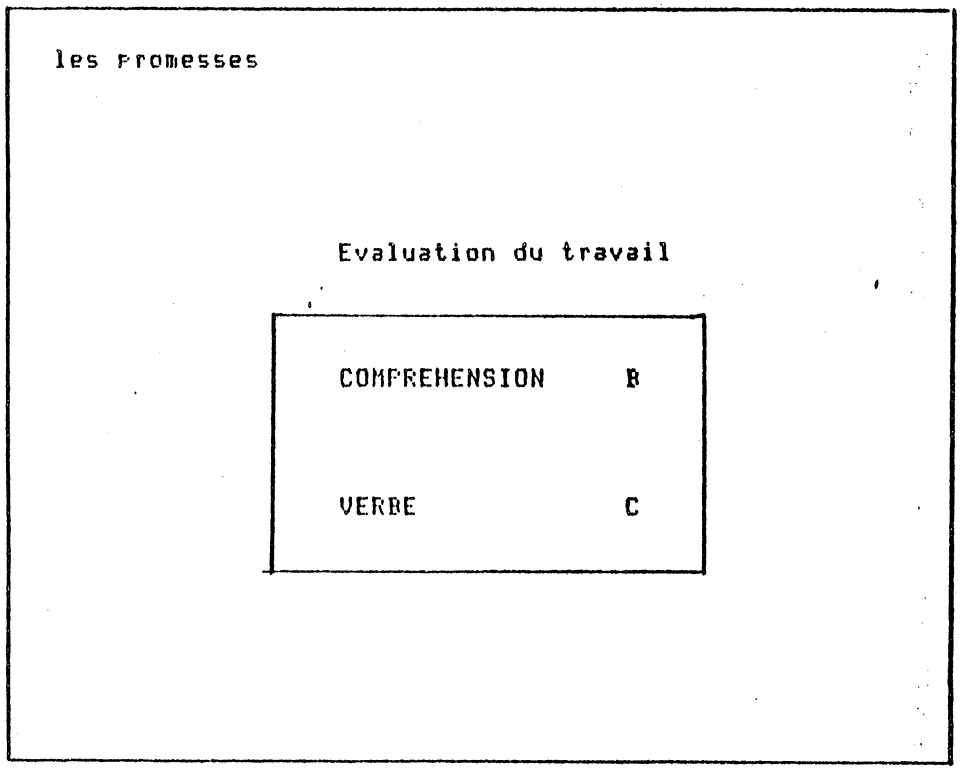
- l'apprenant peut appeler la requête S (secours), dont le contenu s'affichera, dans ce cas précis, dans la partie basse de l'écran.

A la fin,

- l'apprenant écoute le texte complet, lu en français sur cassette.

Une brève évaluation de son travail est donnée à l'apprenant, sous forme d'une notation par lettres.

L'exercice est terminé.



Puis la liste des exercices du FUTUR réapparaît sur l'écran.

- l'apprenant choisit, par exemple,

d'arrêter de travailler.

Liste des exercices du FUTUR

0 arrêt

1	axe du temps	11	puzzle 3
2	si la pollution continue	12	futurs irréguliers
3	puzzle 1	13	que sera l'avenir ?
4	formation du futur	14	puzzle 4
5	les terminaisons du futur	15	bulletin de la météo
6	les promesses	16	proverbes
7	puzzle 2	17	puzzle 5
8	présent ? passé ? futur ?	18	
9	les vacances	19	
10			

Donnez le numéro de l'exercice choisi : 0

La liste des commandes disponibles s'affiche.

- l'apprenant arrête la session de la même façon (en tapant 0).

Le didacticiel rend le contrôle au système.

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU le rapport de présentation de Monsieur P.C SCHOLL, Professeur

Monsieur Jean-Michel ADAM

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 23 novembre 1983

Le Président de l'INP-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,