Novembre 82 N° 7

l'ADELIEN

BULLETIN de l'ADELI

ASSOCIATION LOI 1901 - 33, AVENUE DES GOBELINS, 75013 PARIS - TÉL. : 336 49-19

TABLE DES MATIERES

NOS OBJECTIFS POUR 1982-1983

Pour l'année 1982-1983, le Comité de l'ADELI a élu le BUREAU suivant :

Président :	François TETE Banque WORMS	266 90 10
Trésorier :	Henri BARADAT AFPA Créteil	336 49 19
Secrétaire :	Alain COULON SCAT CII-HB	358 93 35

Rédacteur de l'ADELIEN : Philippe GAUCHET
APSIDE

Relations extérieures : Jean Claude POUILLY
AIR FRANCE

Notre programme comporte les objectifs suivants :

- rencontre du Bureau avec J.D. WARNIER pour envisager une collaboration
- présence de l'ADELI aux manifestations professionnelles (convention informatique, JIIA,...)
- organisation d'une réunion publique sur les normes AFNOR (après leur diffusion) par J.C. UTTER
- réalisation d'une plaquette promotionnelle sur les activités de l' ADELI
- prise de contact avec l'Agence de l'Informatique en vue d'une collaboration technique et financière
- développement de l'ADELIEN
- constitution d'une bibliothèque d'ouvrages techniques relatifs à la logique informatique
- poursuite des travaux de commissions
  Système
  Programmation transactionnelle

Cette liste est ouverte aux suggestions des ADELIENS

La participation de chacun d'entre vous est nécessaire à la réalisation de ces objectifs.
N'hésitez plus à prendre contact avec nous.

Le Bureau
de l'ADELI

CONCLUSIONS DE LA COMMISSION LCP TEMPS REEL

1 - OBJECTIFS DE LA COMMISSION

L'apparition de nouvelles techniques informatiques a conduit de nombreux programmeurs, quelquefois encouragés pas certains enseignants, à délaisser la programmation logique WARNIER (au profit de procédures spatio-temporelles) dans leurs travaux en temps réel.

L'ADELI s'est émue de cette épidémie et a confié à une Commission l'étude des particularités de ces traitements.

Les travaux de cette Commission, à laquelle ont participé J.P. GENESTIER, R. BROSSE, B. BARTHUET, L. LEVY, A. COULON conduisent à énoncer quelques recommandations essentielles.

2 - DEFINITION D'UN VOCABULAIRE

Afin de permettre un dialogue positif entre les membres de la commission, il est indispensable de formaliser un vocabulaire, accessible aux praticiens des systèmes conversationnels, actuellement commercialisés par les différents fournisseurs de logiciels.

2.1 - Transaction-Utilisateur

Ensemble de traitements, traduisant un dialogue relatif à une fonction définie par l'Utilisateur (exemple : une saisie de commandes).
Une Transaction-Utilisateur se décompose en échanges.

2.2 - Echange

Ensemble de traitements qui commence par la reception d'un message introduit par le terminaliste et qui se termine par l'envoi d'une grille vers le terminaliste.

2.3 - Conversation

Laps de temps entre 2 échanges d'une même Transaction-Utilisateur

2.4 - Transaction-Système (ou Routine)

Unité de programmation obtenue par :

- découpage d'un échange volumineux ou
- regroupement d'échanges de petite taille.

3 - PARTICULARITES DE LA PROGRAMMATION CONVERSATIONNELLE

3.1 - Nature des composants du Terminal

Le clavier est un organe d'entrée de données. L'écran est un organe de sortie de données émises par l'unité centrale.
L'affichage, à l'écran, des données introduites au clavier du terminal est un transfert technologique que la programmation spécifique n'a pas à gérer.

3.2 - Les données d'entrée ne préexistent pas à l'exploitation

La séquence des données d'entrée ne peut pas être perturbée lors d'un traitement par lot.
En revanche, au cours d'un traitement conversationnel, c'est le programme qui, en fonction des résultats des contrôles effectués sur les données précédentes, attribue leur signification aux données suivantes.

3.3 - Contraintes des systèmes transactionnels

Pour soulager la programmation, de nombreux systèmes transactionnels (TDS, CICS, TPS, DTF,...) génèrent les traitements technologiques indispensables à la facilité de l'exploitation (sécurités nécessaires aux reprises).

Très fréquemment, ces systèmes imposent un découpage de la Transaction-Utilisateur en Transaction-Système (Routines).

4 - INFLUENCES DES SYSTEMES TRANSACTIONNELS (LOGICIELS DE BASE)

Trois familles de systèmes transactionnels on été distinguées.

4.1 - Systèmes non réentrants

Chaque terminaliste travaille sur la copie du programme. Il n'y a pas de découpage et on écrit un programme pour chaque Transaction-Utilisateur.

4.2 - Systèmes réentrants

a - avec gestion de l'enchainement des Routines.

En principe, chaque Routine (Transaction-Système) correspond à un échange (une réception - un affichage).

b - sans gestion de l'enchainement des Routines.

Chaque programme traite une Transaction - Utilisateur, mais chaque affichage provoque une interruption. Le programmeur doit mémoriser, sous forme d'indicateur, l'emplacement de l'interruption. Lors du retour à cette Transaction - Utilisateur, le programme devra analyser l'indicateur pour connaitre le nouvel échange à exécuter.

5.1 - LCP doit s'appliquer à la totalité de la Transaction-Utilisateur

Il s'agit de satisfaire les besoins des Utilisateurs et non d' optimiser l'emploi des ressources liées à un outil !
Il faut construire :

- le FLS de la Transaction-Utilisateur,
- l'obtention des données internes,
- le FLE des données à utiliser.

Ensuite, si des contraintes de délais sont impératives, il est possible de réorganiser les entrées et les sorties de façon à minimiser le nombre des échanges. Il est souvent bénéfique de regrouper la saisie de plusieurs rubriques en un seul écran.

5.2 - Dans le cas de systèmes réentrants,

Il importe de déterminer soigneusement les échanges (une reception - un affichage)
Lorsque l'enchainement des Routines est assuré par le système transactionnel (4.2-a), il y a intérêt à réaliser une Transaction-Système par échange.

Tout regroupement de plusieurs échanges en une Transaction-Système, doit être justifié par une argumentation quantitative.

Dans certains cas, une structure répétitive complète doit être envisagée

Il arrive fréquemment, en programmation temps réel, que l'on traite des ensembles dont le dernier élément appelle l'utilisation de données distinctes de celles utilisées par ses prédécesseurs.
Dans ce cas, l'emploi normalisé de la structure répétitive complète abordée dans l' Adelien No 6 (la structure répétitive traditionnelle est-elle suffisante ?) semblerait devoir s'imposer.

DEFENSE DE LA STRUCTURE REPETITIVE SIMPLE

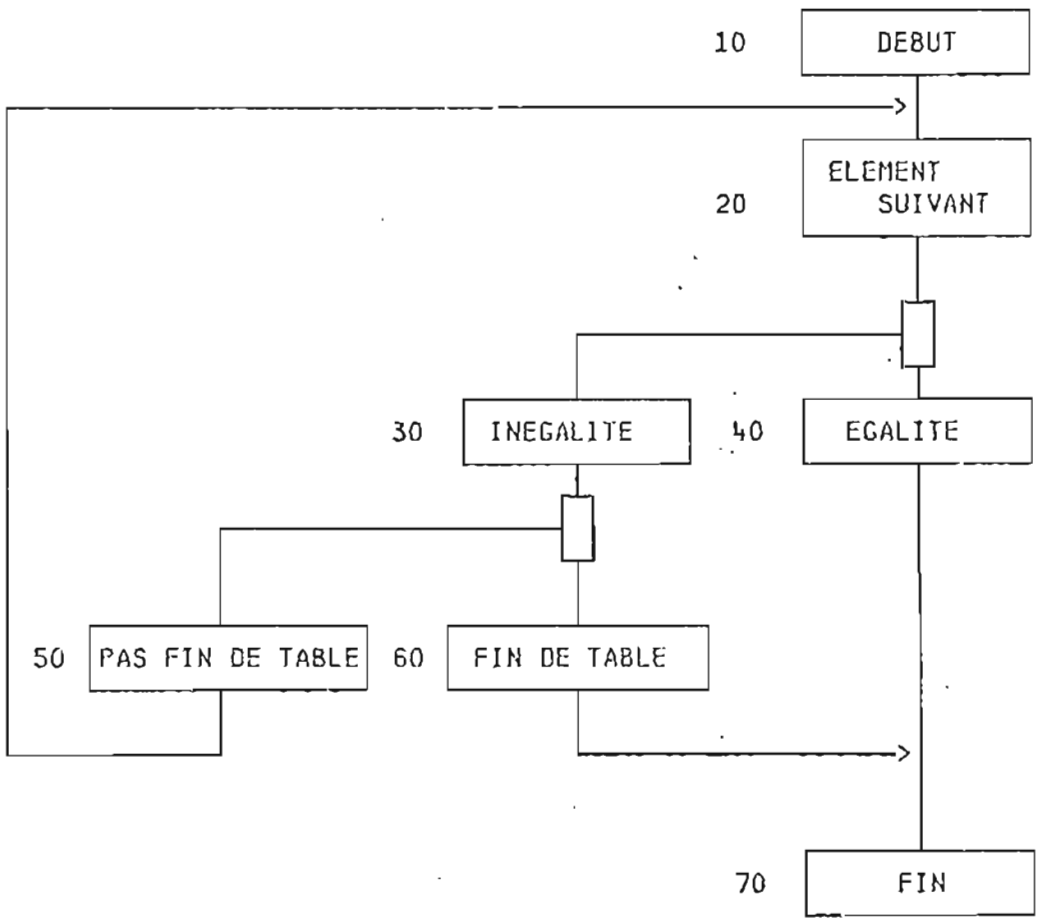
A la question qu'a posée mon ami Alain COULON dans l'article publié dans le numéro 6 de l'Adelien d'Avril 1982. je répondrai que la Structure Répétitive Traditionnelle est largement suffisante et qu'elle se révèle toujours bien adaptée pour traiter le cas général de Répétitive.

Mais pour parler de la Répétitive encore faut-il savoir la définir correctement. Or , l'auteur aux pages 3, 4, 6 a tout simplement défini en compréhension des contenants et non des contenus.

En Logique Informatique, on ne s'intéresse pas à des contenants comme les éléments d'une Table qui sont identiques et de même structure mais au contenu de ces éléments qui sont des Données concernant des Clients, des Salariés ou des Produits.
Or, si l'on traite des Données Employés, l'élément Fin de Table (s'il existe) ne rentre pas dans le cadre de la définition en compréhension des Données Répétitives et sera donc traité à part.

L'auteur, incluant dans sa définition en compréhension l'élément Fin de Table identique aux autres éléments a besoin pour le traiter d'utiliser une Alternative; ceci le conduit par un raisonnement exact à proposer des structures hybrides ni répétitives ni alternatives qui nous ramènent au bon vieux temps de la programmation "Sauvage".

Pour mieux illustrer mon propos, dessinons l'organigramme du programme page 6.



1 - QUELQUES REMARQUES

- La Répétitive LCP peut aussi traiter des ensembles de données non ordonnés.

- Aussi bien dans le FLS, le FLE que le programme, l'auteur nous parle d'articles, d'éléments, de lignes, d'enregistrements qui sont l'aspect physique de contenants. Or, la Logique Informatique est indépendante des techniques, donc des supports physiques.

- La résolution (page 6) du cas général de détection et du traitement du dernier élément FIN (qu'il existe ou non) est toujours possible, simplement avec la Répétitive seule et non par l'utilisation d'une Alternative.
  Il n'y aura donc pas (page 4) des "Séquences parasites (logiquement vides) parfaitement inutiles alourdissant etc...".D'ailleurs en page 5 l'auteur nous signale que "la condition de répétition en 65 est identique à celle de l'exécution de la séquence 40 Fin de Table Alternative.

2 - EXEMPLE BIEN CONNU

Pour résoudre le problème de la recherche séquentielle faut-il encore qu'il soit bien posé. Il faut donc que la Description de l'Ensemble des Données à obtenir (EDO, Sorties, FLS) soit correcte.

L' auteur nous propose deux niveaux de décomposition. Un premier niveau élément dont j'ai déja parlé. Un second niveau pour un élément pour lequel existe trois éventualités de sortie ce qui pourrait nous donner (e) éventualités de sortie puisqu'il y a (e) éléments.

En sortie il n'y a que deux 2 éventualités possibles et exclusives lorsqu'on recherche séquentiellement dans une table de N éléments des données correspondant à une valeur V.

- Ou bien l'on trouve les données correspondantes à la valeur V et on peut les utiliser, les imprimer ou les affecter sur un écran.

- Ou bien l'on ne trouve pas dans la Table les données correspondantes à la valeur V et on le signale par un message.

  Il est donc absolument faux de dire que le résultat que l'on attend de cette recherche est répétitif puis alternatif.

2.1 - ENSEMBLE DE DONNEES A OBTENIR.

(EDO, SORTIES, FLS)

Ensemble des Données à obtenir
- Données correspondant à la valeur V non trouvées dans la Table (0,1)
- (+)
- Données de la Table correspondant à la valeur V (0,1)

Je suis d'accord avec l'auteur pour mentionner sur le FLS les conditions d sortie.

2.2 - ENSEMBLE DES DONNEES A UTILISER.

(EDU, ENTREES, FLE)

Les Données que nous allons utiliser concernent non pas un élément qui est u contenant mais le <u>contenu</u> de cet élément par exemple, des Données concernant U Employé au premier niveau.

Pour UN Employé, je dois savoir si la valeur V (par exemple un numéro d'employé que je cherche correspond à l'Employé que je suis en train de traiter. Pour cela je vais utiliser la valeur V et la comparer au numéro d'employé.

Ensemble des Données à utiliser
- UN EMPLOYE (e)
  - NUMERO EMPLOYE = V (0,1)

Si dans la table à N éléments contenant des Données concernant des Employés j ne trouve pas d'Employé dont le numéro correspond à la valeur V, il m'aura fall explorer tous les éléments de la table donc, tous les Employés.

Si je trouve l'Employé que je cherche, il est inutile d'utiliser les Donnée concernant les autres Employés restants.

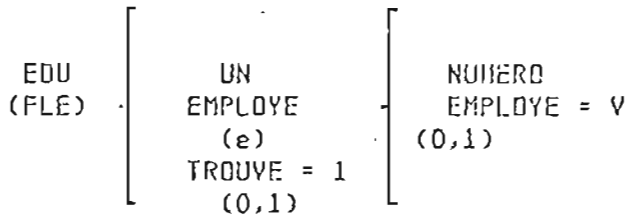
Dans les deux cas, il faut arrêter la répétitive. Il suffit pour cela de comparer :

N = Nombre d'Employés dans la Table
et i = Nombre d'Employés Utilisés

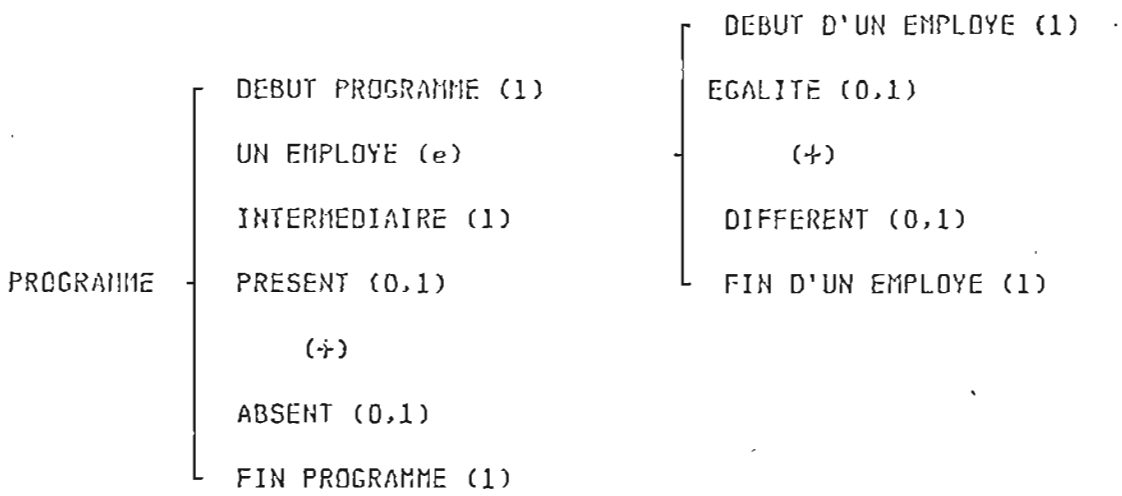
Quand i sera égal à N, la répétitive s'arrêtera. Si je trouve l'Employé que je cherche, je transfère la valeur de N dans i. J'incrémente de un la valeur i à chaque employé après l'avoir initialisé à zéro.

Ce calcul de i est la création d'un critère d'identification d'UN Employé, une donnée secondaire qui n'existe pas en Entrée.

Mais l'étude des Données à Utiliser n'est pas terminée. Il faut que je sorte le message ou les Données EMPLOYES. Pour cela, je n'ai aucune donnée à utiliser. Il faut donc la créer. Je l'appellerai TROUVE. Cette Donnée Secondaire TROUVE sera positionnée à UN (1) si l'on trouve une donnée Employé relative à V, sinon elle restera à sa valeur initiale zéro. La donnée secondaire TROUVE sera utilisée en étant comparé à UN une seule fois.

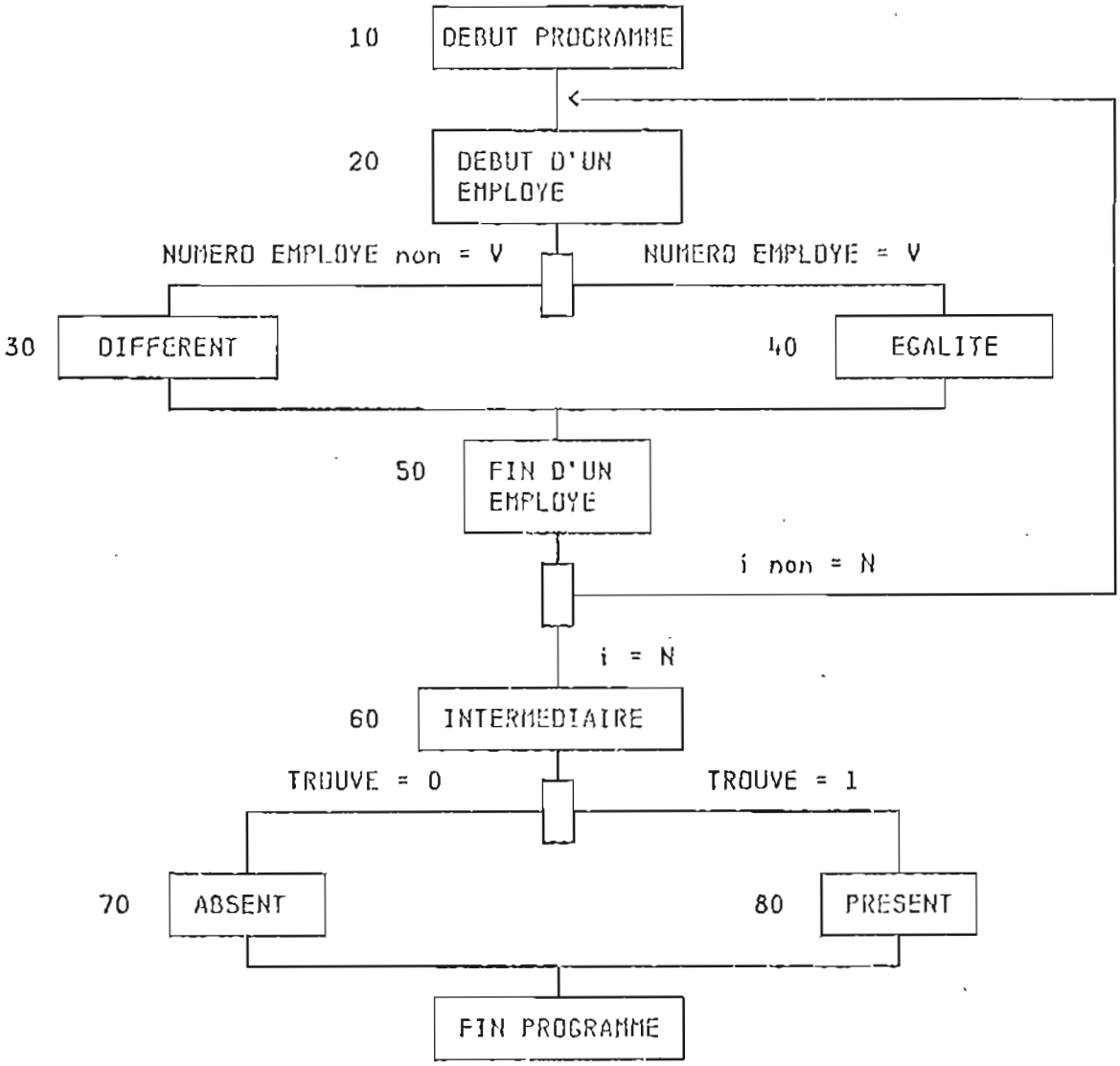


2.3 - LE PROGRAMME.



2.4 - L'ORGANIGRAMME.

Pour être plus visuel et mieux compris.



Quelques Instructions

```
En 10  | i = 0                 En 20  - i = i+1
       | TROUVE = 0               70  - SORTIE MESSAGE
   40  | i = N                    80  - UTILISATION DONNEES
       | TROUVE = 1
```

3 - CONCLUSION

Cette solution comporte une répétitive et une alternative au premier niveau et une alternative au second niveau.

Pour deux microsecondes de plus (2 tests exécutés en plus), à une solution peu compréhensible donc difficilement modifiable, compliquée donc délicate à concevoir, utilisant des structures hybrides, je préfère substituer une solution logique, claire, compréhensible donc facile à modifier, simple donc facile à concevoir, en n'utilisant que les deux structures simples de LCP, la Répétitive et l'alternative.

Henri BARADAT

LOGICAL (D'INFOTEL).

LOGICAL est un ensemble de programmes développés par INFOTEL pour aider l'informaticien dans ses tâches de conception et de réalisation.

LOGICAL comprend deux sous-ensembles complémentaires :

* LOGICAL-ANALYSE, dont l'objet est le passage des spécifications, de l'application à la conception des bases de données relationnelles.

* LOGICAL-PROGRAMMATION, outil conversationnel de conception assistée pour des programmes structurés LCP.

LOGICAL PROGRAMMATION permet une conception progressive et conversationnelle d'un programme en suivant les étapes de la méthode LCP, et en fournissant, pas à pas, la documentation graphique aidant l'informaticien dans la formulation de son problème.
Les données structurées sont introduites en conversationnel au moyen de l'éditeur de texte de la machine de développement.

L'introduction des données du FLS (Fichier Logique de Sortie) permet de visualiser sa structure sous forme graphique (accolades).
L'introduction des données du FLE (Fichier Logique d'Entrée) est ensuite l'étape essentielle puisque, c'est à partir de là que LOGICAL produit automatiquement :

- Le graphique du FLE

- Le graphique du programme par référentiel ou par nom de paragraphe COBOL (accolades LCP).

- La structure du programme en pseudo-langage structuré

- L'ensemble des noms de paragraphes et des instructions de rupture de séquence du programme COBOL

- L'organigramme du programme.

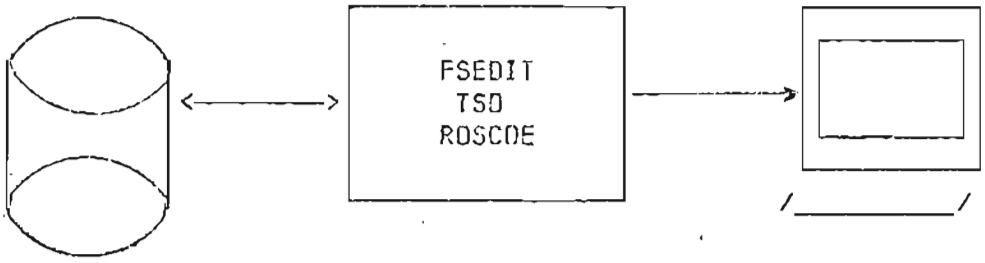
Les instructions par catégorie sont directement placées dans les séquences logiques appropriées, le programme étant alors prêt pour la compilation.

En phase de maintenance, LOGICAL aide l'informaticien en lui produisant, à partir du programme, une documentation à jour, en facilitant l'introduction de nouvelles instructions et en simplifiant la restructuration du programme, tout en maintenant en permanence une documentation à niveau.

Développé en COBOL ANS sur IBM Série 1, LOGICAL PROGRAMMATION sera disponible en septembre 1982.

LOGICAL

SCHEMA DE FONCTIONNEMENT



UN FICHIER UN EDITEUR DE TEXTE UN ECRAN

```
---------- LOGICAL ------------ MENU PRINCIPAL ------INFOTEL------

ENTRER LE PARAMETRE CHOISI                          TAPER PF3 POUR FIN

       VOTRE OPTION ====>

             1 - EDITEUR DE TEXTE
             2 - TRACE DE LA DOCUMENTATION
             3 - GENERATION DE STRUCTURE
             4 - TRACE D'ORGANIGRAMME
             5 - GENERATION DE LA PROCEDURE
             6 - LISTE DES INSTRUCTIONS
             7 - GENERATION DU PROGRAMME
             8 - COMPILATION ALLOCATION EXECUTION
             9 - MAINTENANCE


LL       OOOOOO   GGGGGGG  IIIIIIII  CCCCCCC      AAA     LL
LL      OOOOOOOO GGGGGGGGG IIIIIIII CCCCCCCCC    AAAAA    LL
LL      OO    OO GG           II    CC          AAA AAA   LL
LL      OO    OO GG   GGGG    II    CC         AAAAAAAAA  LL
LLLLLLL OOOOOOOO GGG   GGG IIIIIIII CCCCCCCCC  AAAAAAAAA  LLLLLLLL
LLLLLLL  OOOOOO   GGGGGGG  IIIIIIII  CCCCCCC   AAA   AAA  LLLLLLLL
```

DOCUMENTATION

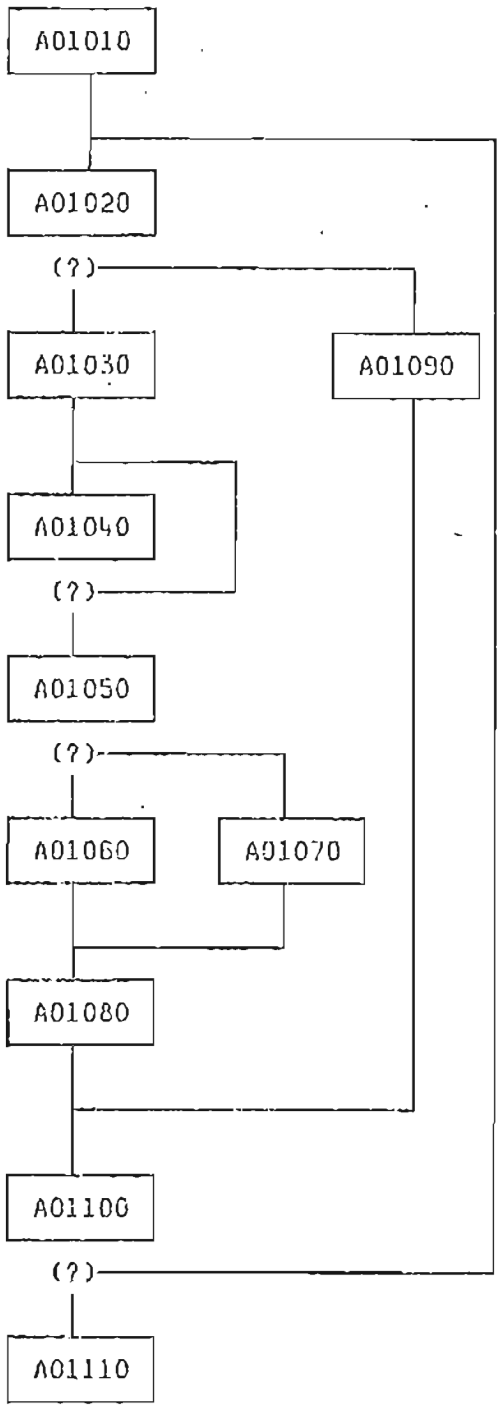
```
D-A01   (1)
  A01010    D-EMPLOYE
              (1)
               A01020     D-ENFANTS (1)
                            A01030
                          ENFANT    (E)
                            A01040
EMPLOYE       ENFANTS     I-ENFANTS (1)
 (EM)          (0,1)        A01050
                          SORTIE    (0,1)
                            A01060
              -ENFANTS    -SORTIE   (0,1)
               (0,1)        A01070
                A01090    F-ENFANTS (1)
              F-EMPLOYE     A01080
               (1)
               A01100

F-A01
  A01110
```

PSEUDO-CODE

```
      A01
DO     D-A010
UNTIL    NOT EFIN
   DO    D-EMPLOYE
   IF       ECODE NOT = "2"   THEN
     DO     D-ENFANTS
     UNTIL     ECODE = "2"   DO ENFANT
     DO     I-ENFANT
     IF     NB-ENFANTS > "2" THEN SORTIE
     ELSE
     DO     - SORTIE
     DO     F-ENFANTS
   ELSE
   DO     - ENFANTS
   DO     F.EMPLOYE
DO     F-A010
```

ORGANIGRAMME



SOFTPEN (D'IPI).

SOFTPEN, "le crayon du logiciel", se veut un outil de spécification et de conception du logiciel.
C'est un outil interactif et graphique s'appuyant sur des concepts respectant à la lettre la Logique Informatique de J.D Warnier.

FONCTIONS ASSUREES :

SOFTPEN permet un suivi complet de la démarche LCP (du FLS jusqu'aux listes par catégories; l'UT se déduisant automatiquement du FLE développé).

L'atout majeur de SOFTPEN, outre la visualisation en interactif des FLS, FLE, UT est de garantir la tenue à jour automatique de la documentation des programmes quelles que soient les modifications apportées (ajouts, modifications ou suppressions de structure).

La maintenance se fait sur les listes par catégories et les modifications sont respectées dans le programme objet. Les temps de réalisation sont ainsi réduits du fait de l''automatisation de la liste ordonnée bien-sûr, de la génération automatique de structure de contrôle COBOL ou PASCAL et surtout de la facilité de correction en mise au point.

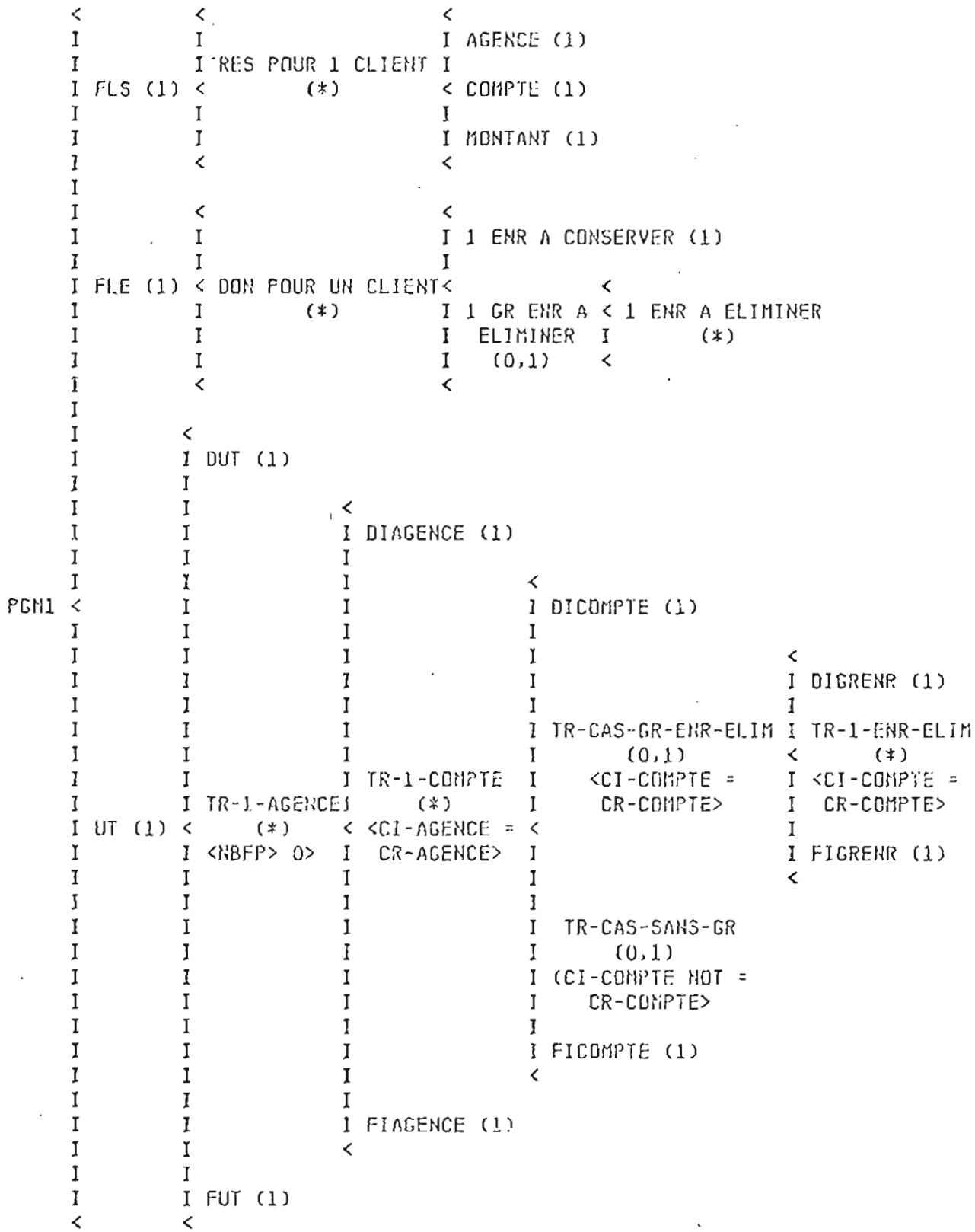
Grâce à la technique des co-programmes, SOFTPEN est aussi bien adapté aux traitements interactifs qu'aux traitements par lots.

PROLONGEMENTS :

Déjà des extensions sont en cours : génération automatique de jeux d'essais, adaptations à LCS, contrôle croisés entre FLS, FLE et UT.

Deux versions sont opérationnelles : - sous IBM VM/CMS et sur VAX/VMS.
Les personnes intéressées peuvent téléphoner à M. Levy pour convenir d'un rendez-vous (Société IPI, 274 32 11).

Exemples de résultats de SOFTPEN



PROGRAMME CORRESPONDANT

```
 IDENTIFICATION DIVISION.

 PROGRAM-ID. NOM-DU-PROGRAMME.
 AUTHOR. NOM-DE-L'AUTEUR.
 REMARKS. DONNEES GENERALES.
 ENVIRONNEMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
 DATA DIVISION.
 FILE SECTION.
 WORLING-STORAGE SECTION.
 PROCEDURE DIVISION.
*
*
*
 STRUCTURE-PGM SECTION.
*---------------------*
*
*
      PERFORM OUT
*
      PERFORM TR-1-AGENCE
      UNTIL NOT
      (NBFP > 0).
*
      PERFORM FUT
*
*
*
 TR-1-AGENCE SECTION.
*-------------------*
*
*
      PERFORM DIAGENCE
*
      PERFORM TR-1-COMPTE
      UNTIL NOT
      (CI-AGENCE = CR-AGENCE).
*
      PERFORM FIAGENCE
*
*
*
 DIAGENCE SECTION.
*----------------*
*
      MOVE NO-AGENCE TO REF-AGENCE.
      MOVE ZERO TO CTR-AGENCE.
*
*
*
```

```
 TR-1-COMPTE SECTION.
*-------------------*
*
*
      PERFORM DICOMPTE
*
      IF
      (CI-COMPTE = CR-COMPTE)
         THEN
              PERFORM TR-CAS-GR-ENR-ELIM
*        ELSE
              PERFORM TR-CAS-SANS-GR.
*
      PERFORM FICOMPTE
*
*
*
 TR-CAS-GR-ENR-ELIM SECTION.
*--------------------------*
*
*
      PERFORM DIGRENR
      PERFORM TR-1-ENR-ELIM
      UNTIL NOT
      (CI-COMPTE = CR-COMPTE).
*
      PERFORM FIGRENR
```

SUR LE MODELE RELATIONNEL

COMMUNICATIONS DE L'ACM DE FEVRIER 1982

Deux articles intéressants ont été publiés récemment dans les communications de l'ACM (Association for Computing Machinery).

Dans le numéro de février 1982, un texte d'Edgar F.CODD, inventeur du modèle relationnel, qui a reçu le prix annuel de l'association américaine pour la contribution majeure à l'informatique.
CODD justifie dans cet article, le modèle relationnel en l'opposant au modèle CODASYL auquel il reproche sa complexité : la structure propriétaire-membre du modèle CODASYL combine, en effet, en une seule construction, trois concepts différents : une liaison 1 à n, une relation de dépendance, et un lien de chainage, qui doit être connu des programmeurs d'application.
C'est ce dernier point qui, d'après CODD, rend difficile son utilisation par les programmeurs, et présente un obstacle insurmontable pour les utilisateurs.

Le modèle relationnel, au contraire, procure une interface claire entre les aspects physiques et logiques.
Il est simple, compréhensible à la fois par les utilisateurs et les informaticiens, et peut être manipulé par des opérations sur des ensembles, (telles que la sélection, la projection et le joint).

En conclusion, CODD affirme que ce modèle relationnel est la base de la productivité pour les applications futures.

Cet article intéressera les concepteurs LCS qui pratiquent ainsi l'approche relationnelle, aboutissant à des bases de données en 3ème forme normale de CODD.

SUR L'AVENIR DE LA PROGRAMMATION

COMMUNICATIONS DE L'ACM DE MARS 82

Dans le numéro de mars 1982, WASSERMAN et GUTZ s'interrogent sur l'avenir de la programmation.

Le logiciel représente souvent plus de 80 % du coût de l'informatique, les dépenses de main d'oeuvre en constituant la part essentielle. Il faut donc contrôler ces dépenses tout en rendant le logiciel meilleur et plus fiable.

Quelles sont les améliorations possibles ?

Les auteurs distinguent trois époques : le court terme, couvrant les cinq prochaines années, le moyen terme, jusqu'à la fin du siècle, le long terme étant à l'horizon du 21ème siècle. Ils ne veulent pas faire un travail de devin, mais identifier les objectifs de recherche d'amélioration pour le futur.

A court terme, les améliorations doivent venir des outils, plus intégrés dans le processus de développement des applications, de la conception à la réalisation, et mieux liés à une méthodologie de réalisation des applications.
Les langages actuels devront subsister, relayés en partie par des langages non procéduraux (type SQL ou QBE).
D'autres améliorations pourront venir de l'environnement (terminaux ergonomiques, traitement de texte intégré, etc) et de la participation plus importante des utilisateurs, aidés par des facilités de prototypage.

A long terme, les prévisions sont plus difficiles : une "révolution" n'est pas impossible. La nature même de la programmation (et des programmeurs) devrait changer.
Les langages seront pratiqués directement par les utilisateurs, les programmeurs travaillant plus au niveau des spécifications, des langages intervenant alors directement au niveau de la conception, et construisant des programmes à partir de la description des sorties.

La distinction entre programmeur et analyste disparaitra alors, l'informaticien du futur devrait être un analyste et un concepteur, la machine se chargeant des tâches de conversion de l'analyse en programmes exécutables.

BIBLIOTHEQUE COMMENTEE

SUR LA "LOGIQUE INFORMATIQUE"

OUVRAGES DE J.D. WARNIER

- Entrainement à la programmation

  Tome 1 : Construction de programmes (avec B. FLANAGAN)

  Tome 2 : Exploitation des données.

  Ces deux ouvrages présentent les versions initiales de LCP. Ils ont un peu vieilli mais sont précieux pour les nombreux exercices qu'ils contiennent.

- Les procédures de traitement et leurs données

  C'est l'ouvrage de base de présentation de LCP.
  La dernière édition contient de nouveaux développements et extensions en particulier, en ce qui concerne le temps réel.

- Pratique de l'organisation des données d'un système : (LCS)

  Précis LCS décrit les principes de la méthode et leur application au cours d'une étude de cas.

- Guide des utilisateurs du système informatique (LOR)

  Précis LOR consacré à la description logique des résultats demandés, est destiné à améliorer la formulation des spécifications et le dialogue informaticiens-utilisateurs.

- Transformation des programmes

  Des développements et des exercices sur l'évolution des solutions LCP et la construction logique des jeux d'essais.

Tous ces ouvrages sont publiés aux éditions d'organisation.

AUTRES OUVRAGES APPLIQUANT LA LOGIQUE INFORMATIQUE DEVELOPPEE PAR J-D WARNIER

- Construction logique de programmes COBOL par M.KOUTCHOUK (Edition MASSON).

  Cours d'initiation au langage COBOL et à la logique LCP : les éléments de logique et de langage sont présentés en parallèle, et appliqués à la réalisation de programme COBOL.

- COBOL : Perfectionnement et Pratique par M.KOUTCHOUK (Editions MASSON)

  Complément de l'ouvrage précédent, amenant à une connaissance complète du COBOL ANS, et comportant, comme illustration, la réalisation complète en LCP d'une chaine de comptabilité.

AUTRES OUVRAGES PRESENTANT DES METHODES DE PROGRAMMATION "STRUCTUREES"

- BERTINI-TALLINEAU : LE COBOL STRUCTURE : UN MODELE DE PROGRAMMATION

  Présentation du COBOL dit structuré (par utilisation de la programmation par référentiels et de l'instruction PERFORM) et de l'arbre programmatique (l'accolade a tourné d'un quart de tour...)

- M.A JACKSON : PRINCIPLES OF PROGRAM DESIGN (Academic Press)

  Présente la construction de programme à partir de l'organisation des données à l'entrée, selon une approche analogue à celle de Warnier, mais moins rigoureuse. Des Développements intéressants sur les coroutines et le "Backtracking" : à traiter en LCP pour juger des différences.

- Programm design and construction par David A.Higgias (Prentice- Hall).

  LCP en Basic et à la mode américaine.

OU TROUVER CES OUVRAGES (ET D'AUTRES) ?

Quelques adresses de librairies parisiennes où le rayon informatique est bien garni :

- Librairie LAVOISIER : 11, Rue Lavoisier - 75008 PARIS

- Librairie DUNOD : 30, Rue St Sulpice - 75006 PARIS

- Librairie LA NACELLE : 2, Rue Campagne Première - 75014 PARIS

- Librairie des Presses Universitaires : 49, Bd St Michel - 75005 PARIS

- Librairie JOSEPH GIBERT : 26-30, Bd St Michel -75016 PARIS

- Librairie EYROLLES : 61, Bd St Germain - 75005 PARIS

- Librairie GIBERT JEUNE : 27 Quai St Michel - 75005 PARIS
  15, Bd St Denis - 75002 PARIS