MASTER EN SCIENCES INFORMATIQUES

ICE-PC. Étude d'opportunité concernant la réalisation d'un environnement de développement d'applications transactionnelles dans le cadre d'un réseau de PC

Bauchau, Damien

Award date:
1990

Awarding institution:
Universite de Namur

- FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX -
Namur

- INSTITUT D'INFORMATIQUE -

ICE-PC

Etude d'opportunité concernant la réalisation d'un environnement de développement d'applications transactionnelles dans le cadre d'un réseau de PC

- Damien BAUCHAU -

Directeur du mémoire : Prof. R. LESUISSE

Mémoire en vue de l'obtention du diplôme de
Licencié et Maître en Informatique

- Année académique 1989 - 1990 -

A Monsieur R. LESUISSE, Chargé de cours aux Facultés Universitaires Notre-Dame de la Paix à Namur, qui a honoré cette recherche d'une attention constante et qui nous a généreusement dispensé ses conseils,

A la Belgolaise et plus particulièrement,

A Monsieur E. de PATOUL, Directeur du Département Organisation et Informatique, qui a guidé nos investigations en nous aidant à nous repérer dans le champ difficile de cette recherche,

A Monsieur PH. JOURION, Ingénieur Système, qui nous a fait bénéficier de ses connaissances et de son expérience,

A Mesdames C. DECOSTER, B. VREBOS et M. VERMEIRE pour leur disponibilité et leur aide appréciable,

Nous tenons à manifester notre plus profonde reconnaissance.

A mes Parents qui m'ont toujours soutenu et encouragé,

à Philippe et surtout à Isa...

SYNTHESE

Confrontée à des problèmes de croissance des coûts, de saturation à moyen terme et de dépendance excessive de la configuration informatique centrale, la BELGOLAISE envisage de décentraliser certaines applications au profit d'un réseau de PC [2], reproduisant sur les micros l'environnement de développement logiciel "ICE" [3] utilisé sur son système informatique central.

Nous avons donc effectué l'étude d'opportunité de ce projet et mis au point un prototype de cet environnement baptisé "ICE-PC" [8]. Ce prototype utilise les services d'un nouveau produit d'IBM, FBSS [4] qui est une couche logicielle d'extension du DOS [5].

Ce projet sera probablement abandonné entre autres par manque de fiabilité et de portabilité à long terme de FBSS. Toutefois, la BELGOLAISE est prête à réétudier ce projet à partir de PC fonctionnant sous UNIX [6], système d'exploitation réputé plus fiable et plus puissant.

SYNTHESIS

Belgolaise, having to cope with problems of increased costs, saturation at medium term and of excessive dependence on its unique mainframe, is thinking of decentralizing some applications to a PC-network [2], reproducing on the micro-computers the environment of the software development "ICE" used in the central computer system.

We have therefore effected the opportunity study of this project and have developed this environment prototype called "ICE-PC" [8]. This prototype makes use of the services of a new IBM product, FBSS [4], which is a DOS [5] extension software layer.

This project will probably be given up for lack of reliability and long-term portability of FBSS. However, Belgolaise is ready to make a new study of this project, starting from PCs working on UNIX [6], reputed to be a more reliable and powerful operating system.

SYNTHESE

De Belgolaise, die geconfronteerd wordt met problemen van kostenverhoging van verzadiging op middellange termijn van bovenmatige afhankelijkheid ten opzichte van haar enige mainframe en, denkt erover sommige toepassingen te decentraliseren ten bate van een PC-netwerk [2], door op de micro-computers het environment van software-ontwikkeling "ICE" te reproduceren dat op haar centrale computersysteem wordt gebruikt.

We hebben dus de opportuniteitstudie van dit projekt uitgevoerd en een prototype van dit environment, "ICE-PC" [8] genoemd, uitgewerkt. Dit prototype maakt gebruik van de diensten van een nieuw IBM-produkt, FBSS [4], dat een software-laag van uitbreiding van de DOS [5] is.

Dit projekt zal vermoedelijk worden opgegeven bij gebrek aan betrouwbaarheid en draagbaarheid op lange termijn van FBSS. De Belgolaise is echter bereid om dit projekt weer te bestuderen met, als basis, PC's die werken onder UNIX [6], een betrouwbaarder en sterker verwerkingssysteem.

TABLE DES MATIERES

Table des matières

Chapitre 2 : Le système informatique de la Belgolaise

Chapitre 3 : Etude approfondie de l'environnement ICE

Table des matières

Chapitre 4 : Le projet de décentralisation

Table des matières

INTRODUCTION

INTRODUCTION

Afin de faciliter la production d'une quantité importante d'applications le département Informatique de la Belgolaise a mis au point un environnement de développement logiciel baptisé "ICE" [3]; celui-ci est utilisé pour la conception et l'implémentation de l'ensemble des applications informatiques de la Banque.

La charge très importante que représente la masse des opérations de mises à jour des soldes clients induit à moyenne échéance une saturation de la capacité de traitements des informations.

En réponse à ce problème de saturation du système informatique central, nous avons réalisé un prototype simplifié de ce même environnement "ICE" [3]. Celui-ci, baptisé ICE-PC [8] fonctionne sur un réseau local [9] de PC. ICE-PC facilite donc la création d'applications PC qui soulagent le mainframe [1] de certaines tâches qu'il assurait jusqu'alors. Cette solution présente deux avantages appréciables :

1) Une diminution des coûts matériel et logiciel puisque cette solution décentralisée fait l'économie du remplacement ou du dédoublement du mainframe et de la location des logiciels de base (OS, SGBD, ...).

2) Un accroissement de la disponibilité du système informatique global puisqu'en cas d'indisponibilité du mainframe, le réseau de PC peut continuer à assurer la saisie des opérations.

Ce mémoire de fin d'études a pour objet de présenter d'une part, la synthèse des aspects pertinents de l'informatique à la Belgolaise, et d'autre part, l'étude d'opportunité et les développements du projet ICE-PC que nous avons réalisés.

Tout au long de ce texte, nous n'avons pas uniquement raisonné en informaticien, mais nous avons aussi essayé de tenir compte de facteurs non techniques. Cette perpective globale d'évaluation est considérée par des auteurs tel M. J. Earl [ER88] comme fondamentale pour une réflexion d'ensemble du phénomène d'informatisation.

Ce mémoire s'articule de la façon suivante :

Le premier chapitre décrit la Belgolaise. Il est, en effet, important de connaître et de comprendre les particularités d'une organisation si l'on veut saisir l'essence de son système d'information.

Le deuxième chapitre décrit le système informatique de la Belgolaise.

Le troisième chapitre présente l'environnement de développement ICE tel qu'il existe aujourd'hui à la Banque. Le lecteur trouvera notamment une description précise des principes généraux qui le gouvernent, de même que leurs justifications théoriques.

Le quatrième chapitre, expose un projet générique dont le but est de produire une version de l'environnement ICE telle qu'elle fonctionne sur un réseau de PC. Ce chapitre contient aussi une brève présentation de deux solutions concurrentes qui permettent chacune d'atteindre l'objectif du projet générique. Il détaille ensuite l'une de ces deux solutions, à savoir : les spécifications, la conception et l'implémentation d'un prototype d'une version PC de ICE qui serait élaborée à partir d'un nouveau logiciel d'IBM : FBSS [4].

Le cinquième chapitre aborde l'étude d'opportunité du projet exposé au chapitre précédent. On y examine différentes considérations techniques, économiques et organisationnelles relatives à ce projet.

Le sixième chapitre synthétise les chapitres précédents et forme la conclusion de ce mémoire.

> En résumé, la description des enjeux, les justifications théoriques et empiriques, en bref, l'ensemble des considérations qui permettent de déterminer si ICE-PC répond à certains problèmes informatiques de la Belgolaise constitue [8] l'essence de ce texte.

CHAPITRE I : PRESENTATION GENERALE DE LA BELGOLAISE

CHAPITRE I : Présentation générale de la BELGOLAISE

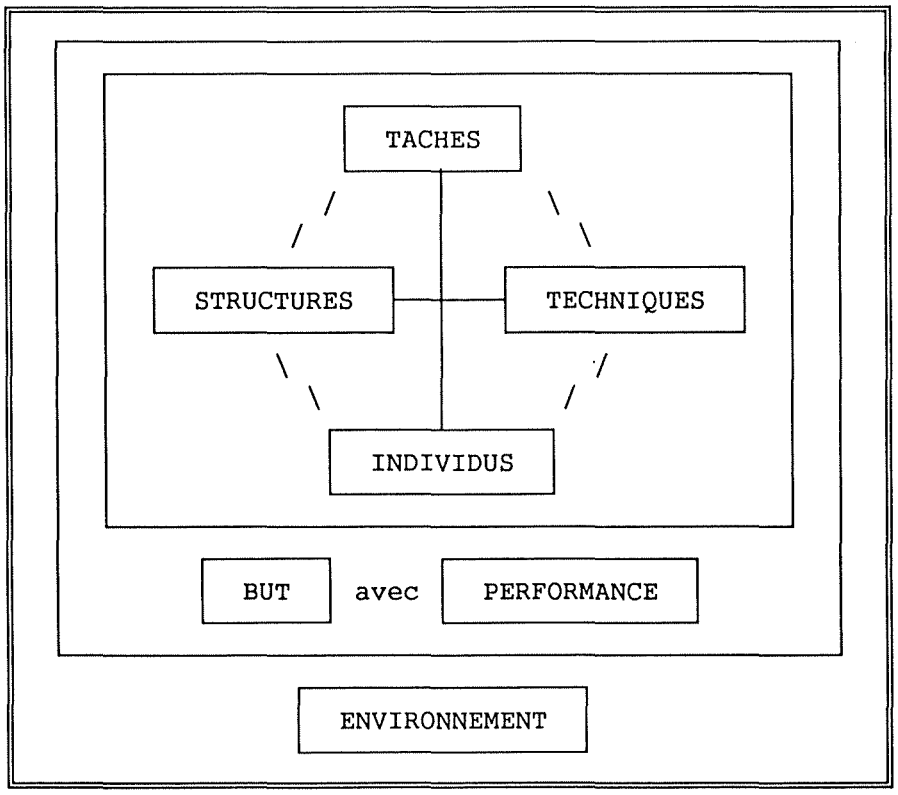
Avant de rentrer dans le vif du sujet, c'est-à-dire d'aborder l'étude approfondie de l'informatique à la Belgolaise et plus particulièrement de l'environnement ICE, le chapitre I décrit avec précision l'organisation dans laquelle cet outil a été mis au point : la Belgolaise.

En effet, il n'est pas possible d'étudier correctement un système d'information sans comprendre l'organisation pour laquelle il doit fonctionner.

Rappelons que selon Leavitt [LV65], une organisation est une configuration particulière de différents éléments :

1) des tâches,
2) des techniques,
3) des individus,
4) des structures.

Ces quatre éléments baignent dans un environnement et tentent d'atteindre par leurs interactions un but avec une certaine performance.



Modèle de Leavitt
Fig. I - 1

Même si ce modèle par sa simplicité est sujet à critiques, choisissons-le comme définition du concept d'organisation et utilisons-le pour nous guider dans la description de la Belgolaise.

Ce premier chapitre s'articule donc selon 7 sections :

La première qui replace l'organisation dans une perspective évolutionniste (EVOLUTION),

la deuxième qui précise les activités spécifiques de la Belgolaise (TACHES),

la troisième qui décrit la structure financière et organisationnelle de la Banque (STRUCTURES),

la quatrième qui caractérise et décrit l'évolution de son personnel (INDIVIDUS),

la cinquième qui cerne le marché sur lequel elle réalise ses activités (ENVIRONNEMENT),

la sixième expose les buts et la stratégie d'affaires de la Banque (BUT - PERFORMANCE),

la septième propose la conclusion de ce premier chapitre.

Le système informatique de la Belgolaise - c'est-à-dire l'exposé de la composante technique (TECHNIQUE) du modèle de Leavitt - est présenté dans le chapitre II.

1. EVOLUTION : La Belgolaise hier et aujourd'hui

La Banque trouve son origine dans la BANQUE DU CONGO BELGE constituée le 11 janvier 1909 sous forme de société anonyme. La BANQUE DU CONGO BELGE, qui avait établi son Siège social à Bruxelles, exerçait l'essentiel de ses activités en Afrique. Par la convention du 7 juillet 1911, la Colonie du Congo Belge confia à la BANQUE DU CONGO BELGE le privilège d'émission pour une période de 25 ans prolongée ensuite jusqu'au 1er juillet 1952. Par la même convention, la Banque s'engageait à assurer le service de la Caisse de la Trésorerie de la Colonie.

Le 10 août 1911 fut constituée la BANQUE COMMERCIALE DU CONGO, dont l'objet était de reprendre les opérations auxquelles la BANQUE DU CONGO BELGE devait renoncer en raison du privilège d'émission qui venait de lui être conféré. Cette Banque fut constituée au capital de 1 million de francs, représenté par 2000 actions de 500 francs. La BANQUE DU CONGO BELGE ouvrit en 1914 un Siège à Londres et en 1919 une Succursale à Anvers. Le 16 septembre 1952, la BANQUE DU CONGO BELGE se transforma en société congolaise par actions à responsabilité limitée et transféra son Siège de Bruxelles à Léopoldville (actuellement Kinshasa).

Le 19 novembre 1952, suite à la déchéance du privilège d'émission, la BANQUE COMMERCIALE DU CONGO et la BANQUE DU CONGO BELGE décidèrent la fusion des deux sociétés.

A partir du moment où le Congo allait devenir un Etat souverain, il devint indiqué de créer une société distincte de droit belge. Cette société, la BANQUE BELGO-CONGOLAISE, fut constituée au capital de 300 millions de francs belges représentés par 175.000 actions sans désignation de valeur nominale. La BANQUE DU CONGO BELGE faisait apport à la nouvelle société de son immeuble du Cantersteen à Bruxelles ainsi que de matériel et de mobilier, en échange de 48.125 actions, et souscrivait en espèces 126.865 des actions restantes.

Le 25 mai 1960, une assemblée générale extrordinaire de la BANQUE DU CONGO BELGE décida à l'unanimité de réduire le capital social de 500 à 200 millions de francs par voie de répartition entre les actionnaires des actions de la BANQUE BELGO-CONGOLAISE à raison d'une action contre quatre actions de la BANQUE DU CONGO BELGE.

Il résulta de la création de la nouvelle Banque que toutes les opérations qui trouvaient leur point de départ au Congo et leur aboutissement en Belgique ou à Londres - ou inversement - et qui, précédemment, étaient du ressort exclusif de la BANQUE DU CONGO BELGE, furent dorénavant traitées par deux établissements entièrement distincts, agissant vis-à-vis de l'autre en collaboration étroite mais en qualité de correspondant.

En 1965, l'appellation "BELGOLAISE" fut adoptée comme dénomination sociale abrégée de la BANQUE BELGO-CONGOLAISE.

A la suite du changement de nom de la République Démocratique du Congo en République du Zaïre par la loi du 29 octobre 1971, une assemblée générale extraordinaire réunie le 26 avril 1972 adopta la dénomination sociale "BANQUE BELGO-ZAIROISE", en néerlandais "BELGO-ZAIRESE BANK".

La dénomination sociale abrégée "Belgolaise" désignant la banque a été maintenue et continue à être employée, soit isolément, soit cumulativement avec la nouvelle dénomination.

2. TACHES : Les activités spécifiques

La Belgolaise exerce toutes les activités traditionnelles d'une Banque. Cependant, ce n'est pas une banque comme les autres. En effet, contrairement aux sociétés qui visent une large clientèle, la Banque Belgo-Zaïroise s'est, dès l'origine, spécialisée dans le financement du commerce et des transactions internationales. Plus particulièrement, elle concentre toute son énergie à la réalisation de crédits documentaires [10] entre l'Afrique et l'Europe (Un crédit documentaire est un acte par lequel les banquiers d'un importateur et d'un exportateur garantissent à leur client respectif la bonne fin d'une transaction internationale (livraison de biens et/ou services)). C'est à ce titre que la Banque possède l'un des plus gros départements de Crédits Documentaires de Belgique.

3. STRUCTURE : Les structures financière et organisationnelle

Cette section doit déterminer la structure de la Belgolaise. Il faut donc d'une part, préciser sa structure externe - c'est-à-dire sa structure financière - et d'autre part, sa structure interne - c'est-à-dire l'organisation de ses services.

3.1. La structure financière

Pour déterminer la structure financière de la Belgolaise, nous proposons d'une part, de préciser quels sont les principaux actionnaires et les principales participations et d'autre part, d'identifier les éléments significatifs des derniers bilans.

La Banque est une filiale de la Générale de Banque. Celle-ci détient un peu plus de 42 % de son capital. Environ 8 % de ce dernier se trouve aux mains d'autres actionnaires stables et enfin le restant, soit presque la moitié des actions est répartie dans le grand public [BL88].

En aval la Belgolaise possède plusieurs participations. Citons, pour mémoire, les trois plus importantes, à savoir la Banque Commerciale Zaïroise, la Banque de Crédit de Bujumbura et la Banque de Kigali.

La figure ci-après met en lumière les éléments significatifs des derniers bilans.

	1989	1988	1987	1986	1985
Total du bilan	70.412	60.634	46.555	49.637	48.997
Fonds propres (1)	3.468	3.264	3.102	2.953	2.702
Dépôts de la clientèle	35.393	31.991	26.572	27.208	27.839
Dépôts des banquiers et filiales	28.372	22.531	14.995	17.035	15.836
Concours au secteur privé belge et international (2)	21.166	17.589	13.323	13.905	14.941
Concours au secteur public belge (3)	15.225	14.881	12.405	12.530	12.335
Bénéfice brut avant amortissements et impôts	1.084	1.005	1.241	1.331	1.449
Bénéfice net de l'exercice (4)	452	434	430	437	421

Eléments significatifs du bilan et des résultats
Fig. I - 2

(1) : Passif non exigible + report à nouveau (après répartition bénéficiaire).
(2) : Débiteurs divers + reports et avances sur titres + effets commerciaux + débiteurs par acceptations (sous déduction des acceptations de la banque en portefeuille) + cautions données pour compte de tiers + effets réescomptés - concours au secteur public belge enregistré en "débiteurs divers".
(3) : Effets publics + réserve légale + fonds publics belges + crédits utilisés par le secteur public belge ou sous sa garantie.
(4) : Après virement, au non exigible, des plus-values immunisées sur ventes du portefeuille-titres.

Fig I.2.a: Evolution du total du bilan

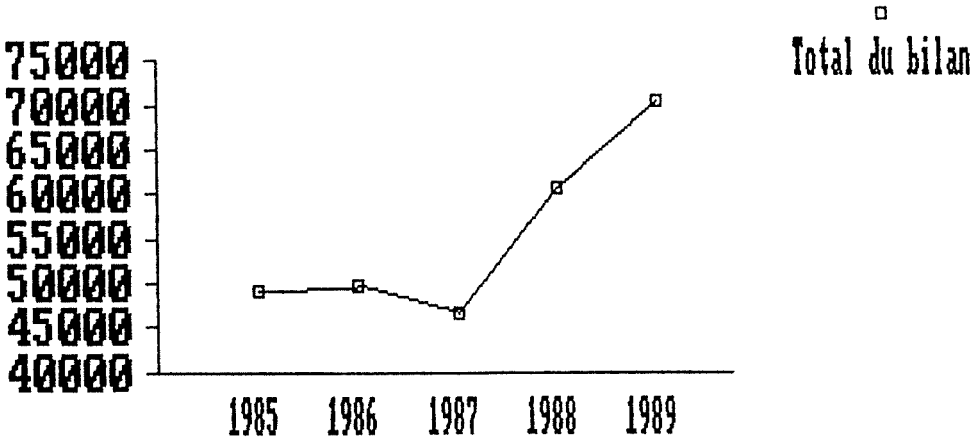


Fig I.2.b: Evolution des fonds propres

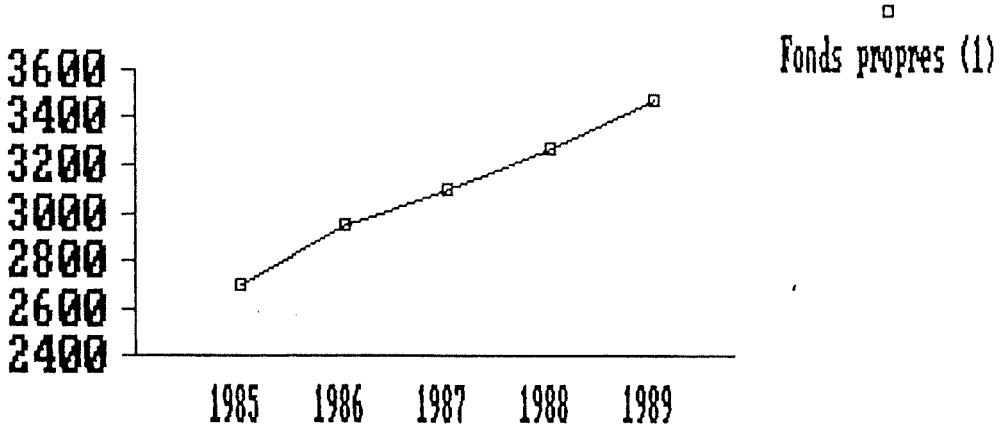


Fig I.2.c: Evolution des dépôts.

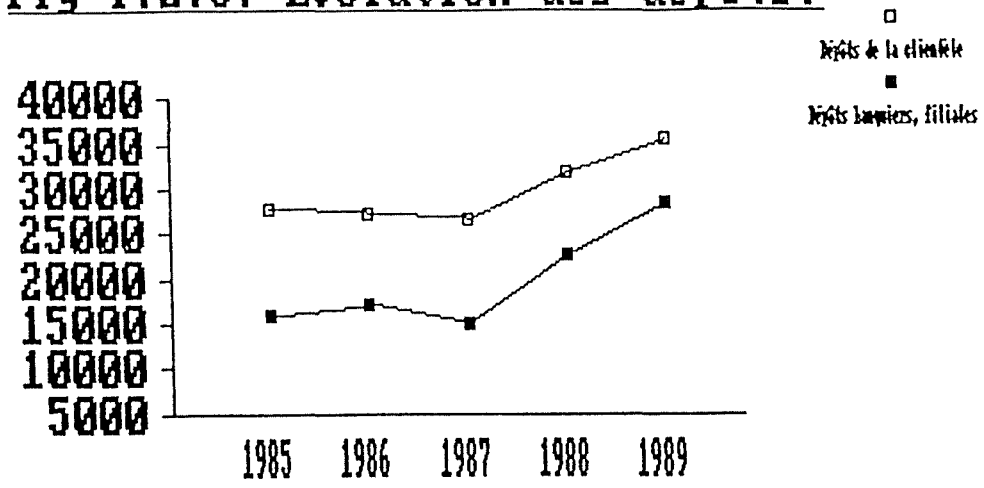


Fig I.2.d: Evolution des bénéfices.

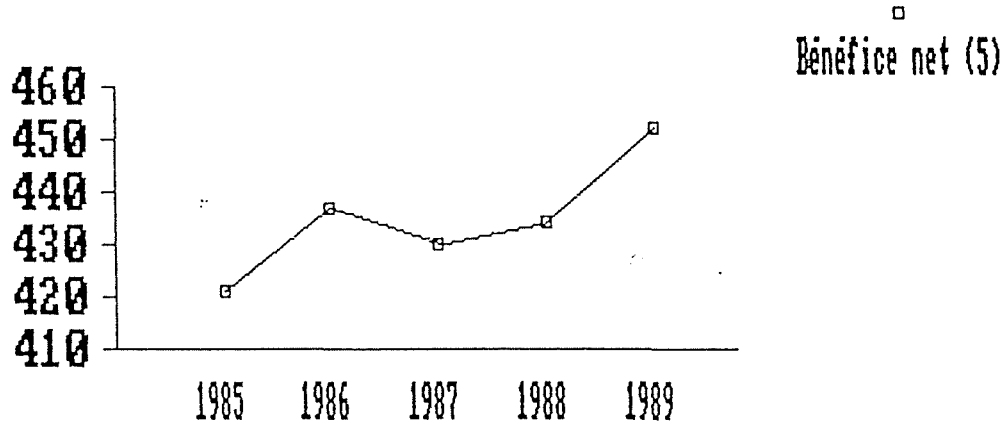
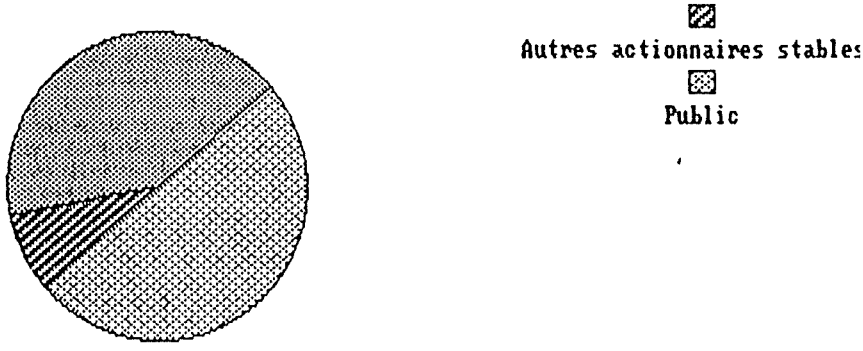


Fig I.2.e : Répartition de l'actionnariat



1988

3.2. L'organisation des services

Le schéma, page suivante, (Fig. I - 3) représente l'organigramme de la société. Le lecteur intéressé trouvera, en annexe, une explication succincte de chacun des services répertoriés. Le "département ressources et matériels informatiques" est développé complètement au Chapitre II.

4. INDIVIDUS : Le personnel

Le tableau (Fig. I - 4) ci-après décrit l'évolution et la ventilation du personnel entre les différents services présentés plus haut.

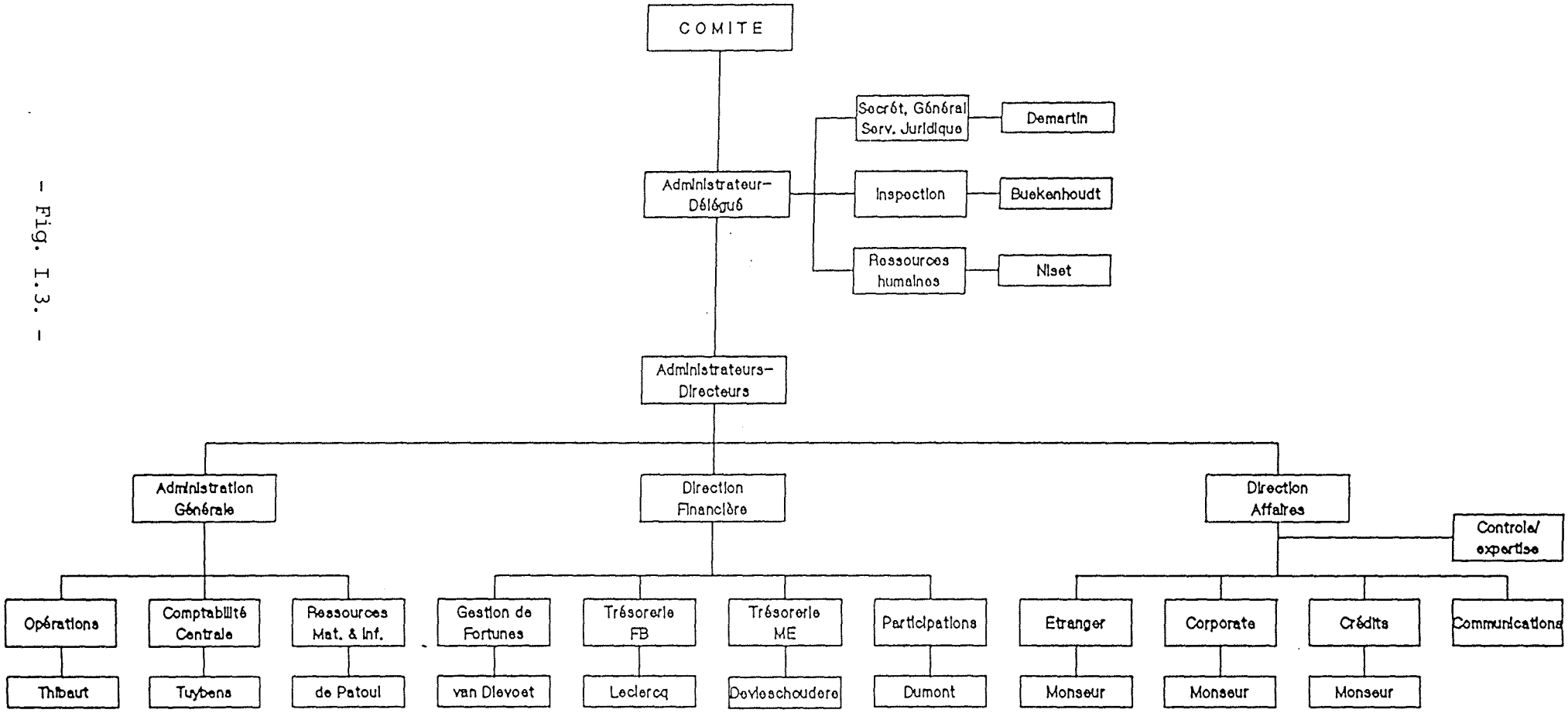
EVOLUTION DU PERSONNEL

	12.82	12.83	12.84	12.85	12.86	12.87	12.88	12.89
Direction/ Secrétariat	3	5	6	6	6	6	6	6
Siège de Bruxelles	264	260	281	277	262	252	253	232
Crédits	33	32	35	38	40	43	49	49
Organisation et Informatique	79	82	79	82	87	92	97	88
Comptable	45	45	48	44	44	42	35	29
Administratif	75	74	81	81	79	75	80	75
TOTAL EMPLOYES	499	498	530	528	518	510	520	479

Sources : [BL88] Fig. I - 4

Constatons tout d'abord que la Belgolaise emploie actuellement pour des fonctions administratives 479 personnes (personnel ouvrier non compris) réparties entre 31 services. L'évolution suit un mouvement globalement ascendant puis descendant. On passe de la sorte de 499 personnes en 1982 à

- Fig. I.3. -


COMITE
Secrét. Général Serv. Juridique
Demartin
Administrateur-Délégué
Inspection
Buekenhoudt
Ressources humaines
Niset
Administrateurs-Directeurs
Administration Générale
Direction Financière
Direction Affaires
Controle/ expertise
Opérations
Comptabilité Centrale
Ressources Mat. & Inf.
Gestion de Fortunes
Trésorerie FB
Trésorerie ME
Participations
Etranger
Corporate
Crédits
Communications
Thibaut
Tuybens
de Patoul
van Dievoet
Leclercq
Devleschoudere
Dumont
Monseur
Monseur
Monseur

479 en 1989. Le maximum est atteint en décembre 1984 où l'on dénombre alors 530 agents. Cette diminution est explicable par l'informatisation systématique des tâches opérationnelles, souvent grosses consommatrices de ressources humaines. Citons en exemple le service comptable qui comptait encore, en 1982, 45 personnes et n'en a plus aujourd'hui que 29.

5. ENVIRONNEMENT : Le marché

Nous avons jusqu'à présent étudié les grandes lignes de l'organisation, sa structure interne, ses caractérisques financières et son personnel. Examinons maintenant la position que la Belgolaise occupe sur son marché. Pour cela, nous proposons d'utiliser le modèle de Porter [PR79].

5.1. Aspect théorique

Porter a mis en évidence un modèle aujourd'hui traditionnel appelé "modèle des forces concurrentielles".

Il existe selon lui cinq forces fondamentales qui interagissent sur l'organisation :

1) la menace des nouveaux entrants,
2) le pouvoir de négociation des fournisseurs,
3) la menace des produits ou services de remplacement,
4) le pouvoir de négociation des clients,
5) les rivalités internes à l'organisation.

L'intensité avec laquelle agira chacune de ces forces serait, selon l'auteur, fonction de la nature et de la structure du secteur industriel concerné [PR79].

5.2. Application à la Belgolaise

Pour ne pas rester uniquement au niveau théorique, nous avons tenté d'appliquer les concepts du modèle de Porter au cas de la Belgolaise. C'est l'objet de cette section que de déterminer quels sont les fournisseurs et les clients de la Banque de même que de présenter ses principaux concurrents. Les aspects des forces internes et des nouveaux entrants sur le marché nous ont semblé peu pertinents dans cette analyse. Pour cette raison, ils ont été laissés de côté.

1) Qui sont les clients de la Banque Belgo-Zaïroise ? Quel est leur profil ? La Société, par son activité particulière [Cfr I - 2.] s'est petit à petit orientée vers une clientèle qui le plus souvent ne réside pas en Belgique mais en Afrique et en particulier au Zaïre, au Rwanda ou au Burundi. En conséquence, environ 80 % des ordres arrivent par courrier. Une part non négligeable de ces ordres parvient aussi par téléphone. Le client est rarement présent à la Banque.

   Il se pose donc depuis longtemps le problème de l'identification des correspondants. Dans ce but, une application quasi unique en Europe a été développée au sein du service Visa [11]. Un programme permet en effet d'authentifier une signature à partir de signatures-types préalablement digitalisées.

   Enfin, signalons qu'environ 10 % des titulaires des comptes sont des entreprises mais que cela représente plus de 25 % du volume des opérations administratives [BL88].

2) Le terme "fournisseurs" est à comprendre comme étant l'ensemble des acteurs qui mettent à sa disposition la matière première sur laquelle l'entreprise réalise des transformations et fixe la valeur ajoutée. Vu sous cette angle, les fournisseurs de la Belgolaise sont aussi ses clients. Ce sont ceux qui lui confient leurs avoirs.

   Les considérations des paragraphes précédents sont dès lors aussi valables pour celui-ci. D'autre part, en se reportant au tableau des éléments significatifs du bilan et des résultats [Cfr I - 3.1., Fig. I - 2], on constate que les dépôts des filiales et banquiers sont en moyenne deux fois moins importants que ceux de la clientèle. Par contre, les prêts se répartissent plus ou moins de façon équivalente entre les secteurs public et privé.

3) Les principaux concurrents sur le marché africain sont les filiales des grandes Banques belges et les Banques locales à capitaux d'Etat. Tandis qu'en Europe et plus particulièrement, en Belgique, les principaux concurrents sont les départements non résidents des grandes Banques qui mènent une politique commerciale pour la catégorie de clients non résidents.

5.3. Conclusions

D'un point de vue théorique, Porter [PR79] conseille aux entreprises de rechercher un avantage soutenu sur leurs rivales en dressant les barrières les plus solides et les plus infranchissables possible envers les concurrents et les nouveaux entrants grâce à l'utilisation judicieuse des technologies de l'information.

D'autre part, il leur conseille aussi d'établir une position forte et profitable par rapport aux deux autres forces concurrentielles : celle des fournisseurs et celle des clients. Il s'agit de renforcer les liens avec les fournisseurs et/ou les clients ou d'en diminuer le pouvoir de négociation à l'aide d'un SI (Système d'Information) approprié.

Enfin Porter et Millar stipulent qu'un tel usage des technologies de l'information peut avoir des résultats aussi impressionnants qu'irréfutables : "La révolution de l'informatique affecte la concurrence de trois façons : elle change la structure de l'industrie et, par le fait même, altère les règles du jeu de la concurrence. Elle crée un avantage concurrentiel en donnant aux firmes de nouveaux moyens de surpasser leurs rivales. Elle donne naissance à de nouvelles opportunités d'affaires" [PR85].

Renforcer les liens avec ses clients c'est entre autres ce que la Banque assure en leur proposant des services spéciaux notamment grâce à l'informatique. Rappelons pour mémoire le système d'authentification des signatures ou encore la possibilité pour un client de disposer chez lui d'un terminal (en Belgique ou à l'étranger) capable de le renseigner en temps réel sur la situation de ses comptes via le service "B-Line" [23]).

Selon Porter, l'entreprise peut faire face à ces forces en choisissant l'une des trois stratégies génériques suivantes : la domination par les coûts (cost leadership), la différenciation ou encore la stratégie de niche (focus). Chacune de ces stratégies, à sa façon, permet de contrer l'ensemble des forces concurrentielles.

De ces différents points de vue, nous dirons que la Belgolaise joue sur deux des trois tableaux. Elle défend une niche particulière, à savoir le financement des transactions commerciales internationales et tout particulièrement celles à destination de l'Afrique.

D'autre part, la principale variable sur laquelle la Banque peut agir est la domination des coûts. Ici, l'informatique joue évidemment un rôle tout particulier. Elle permet de diminuer, par automatisation, les coûts de traitement de la plupart des opérations bancaires, financières et comptables. Si l'informatique à la

Belgolaise est perçue comme un instrument d'intégration des informations et des connaissances, elle est aussi interprétée comme un outil de rationalisation et de diminution des charges d'exploitation.

6. BUT ET STRATEGIE DE LA BELGOLAISE

Nous allons ici brosser en quelques lignes la stratégie d'affaires de la Belgolaise. Le lecteur comprendra facilement que nous ne nous étendions pas sur ce point vu son caractère particulièrement confidentiel.

La Banque s'est, depuis longtemps, positionnée dans une niche spécifique : l'Afrique Centrale. Elle compte s'y maintenir et y accroître sa part de marché [Cfr I - 2.]. Cependant, on discerne actuellement une diversification allant de pair avec l'internationalisation des affaires.

"Pour être rentable une organisation doit créer une valeur totale qui excède les coûts de création et pour obtenir un avantage compétitif sur ses concurrentes, une organisation peut (...) réaliser ses activités à un coût inférieur à celui de ses concurrentes; les technologies de l'information peuvent contribuer à atteindre cet objectif en automatisant certains postes de travail, en intégrant de manière plus étroite des fonctions et sous-systèmes de l'organisation. C'est la manière classique dont les dirigeants ont généralement perçu les projets d'informatisation" [LS89].

De même, la Belgolaise - qui désire conserver son marché africain et ne peut à l'infini augmenter le niveau des ses affaires dans cet espace- doit jouer sur ses coûts. 70 % des frais généraux sont consacrés aux rémunérations du personnel. Comme dans l'ensemble du secteur bancaire, l'entreprise tente dès lors d'optimiser l'utilisation des ressources humaines par compression de la masse salariale.

Pour atteindre cet objectif, deux outils , entre autres, peuvent être utilisés. L'informatique et le recours systématique à la sous-traitance.

La sous-traitance systématique des services généraux, comme l'entretien ou le mess, permet aussi la maîtrise des coûts salariaux, mais son étude dépasse le cadre de ce travail.

L'informatique permet, par l'automatisation des tâches opérationnelles, de rationnaliser bon nombre de services. Citons l'exemple du service de comptabilité dont l'informatisation a permis de réduire le personnel de 40 % [Cfr I - 4., Fig. I - 4].

Inversement, l'automatisation de certains services induit parfois une augmentation importante en terme de personnel. Les gains de productivité sont souvent cachés par les investissements techniques et humains préalables à l'informatisation. Exemple, le département organisation et informatique passe de 79 à 97 personnes en six ans.

Mais si la maîtrise des coûts est une stratégie efficace, soulignons qu'il en existe d'autres. L'entreprise peut aussi "se différencier par rapport à ses concurrentes et mettre à la disposition de l'environnement une offre dont le caractère est unique et reconnu comme tel par celui-ci. En procédant de la sorte, l'organisation étend en fait la chaîne de valeur pour y englober le client et/ou le fournisseur par la création d'un lien électronique. Par exemple les banques développent, à destination de leurs agences, des logiciels susceptibles de leur donner une position concurrentielle : calcul de divers avantages fiscaux liés à des produits financiers, calcul rapide d'octroi de crédits..."[LS89].

Evidemment, la Belgolaise ne possède pour ainsi dire aucune agence, mais elle pourrait cependant monnayer son expérience et sa maîtrise de la technologie informatique à usage bancaire. Par exemple, la Banque a déjà partiellement couvert les frais de développement de son système "B-LINE" par sa revente à un concurrent étranger. Ce système "B-LINE" offre aux clients importants une liaison informatique qui permet l'échange électronique des données [23].

Le plus grand réseau de distribution de produits informatiques est, d'ailleurs, la première banque du pays : la Générale de Banque. Ses agences assurent la promotion des produits du "Direct banking". Visant les PME, la Générale de Banque a pour objectif de fidéliser davantage sa clientèle.

7. CONCLUSION DU PREMIER CHAPITRE

Au terme de ce chapitre de présentation de la Belgolaise, il nous semble important de tirer quelques conclusions.

1) La Belgolaise est une Banque :

   - de taille moyenne,
   - dont l'organisation est : classique et centralisée (pas d'agence).

2) Son activité spécifique réside dans le financement des transactions commerciales internationales et tout particulièrement entre l'Europe et l'Afrique (Crédits Documentaires).

3) La stratégie de l'organisation est essentiellement de :

   - Conserver le leadership sur le marché bancaire africain.
   - Rester compétitive par contrôle des coûts.

4) L'informatique est donc une ressource stratégique car elle permet d'améliorer la compétitivité en diminuant les coûts de traitement des informations grâce à une informatisation des tâches opérationnelles.

Après avoir décrit l'organisation de la Belgolaise, nous abordons dans le Chapitre II la présentation détaillée de son système informatique.

CHAPITRE II : LE SYSTEME INFORMATIQUE DE LA BELGOLAISE

CHAPITRE II : Le système informatique de la Belgolaise

Dans le premier chapitre, nous nous sommes attachés à décrire l'entreprise à l'aide du modèle de Leavitt. Nous avons laissé de côté le caractère technique de l'organisation. Nous abordons maintenant l'examen de son système informatique, c'est-à-dire l'élément TECHNIQUE du modèle utilisé pour décrire la Belgolaise.

Nous proposons de réaliser cet examen à l'aide de deux modèles théoriques : celui de Leavitt [LV65] et celui de Nolan [NL79].

Ce chapitre s'articule donc selon 3 sections :

La première décrit le système informatique de la Belgolaise en termes du modèle de Leavitt.

La seconde décrit le système informatique de la Belgolaise en termes du modèle de Nolan.

Le troisième propose une conclusion de ce chapitre.

1. DESCRIPTION EN TERMES DU MODELE DE LEAVITT

Si nous avons utilisé le modèle de Leavitt pour définir ce qu'est une organisation, nous pourrions utiliser ce même modèle pour préciser la notion de système d'information.

Dès lors, un système d'information sera défini comme une configuration particulière de différents éléments : des tâches, des techniques, des individus et une struture; ces quatre éléments baignent dans un environnement particulier et tentent d'atteindre par leurs interactions un but avec une certaine performance.

Comme dans le chapitre précédent, nous allons utiliser les éléments de ce modèle pour nous guider dans la présentation du système informatique de la Belgolaise.

Cette section consacrée à la présentation du système informatique en termes du modèle de Leavitt s'articule donc de la façon suivante :

- EVOLUTION : l'historique du département informatique.
- TACHES : les activités informatisées.
- STRUCTURE : l'organisation du département informatique.
- INDIVIDUS : le personnel du département informatique.
- TECHNIQUES : l'architecture du système informatique.
- BUTS : les buts et stratégie du système informatique.

Ensuite, nous appliquerons le modèle de Nolan [NL79] à ce département et à l'organisation dont il est un composant essentiel, afin de déterminer à quel stade de son évolution il se situe.

1.1. EVOLUTION : Historique du département

L'année 1968 a vu le remplacement de l'équipement mécanographique classique par un ordinateur IBM 360/20 et le transfert progressif sur celui-ci de l'enregistrement des opérations et de la tenue de la comptabilité.

Dans les premiers jours de 1970, l'unité centrale a été remplacée par une autre plus puissante et plus rapide qui permit une exploitation plus poussée des possibilités de l'ordinateur et la mise en route de nouvelles applications.

En 1978, la Banque est passée à un ordinateur IBM 370/115 muni d'un DOS (Disk Opérating System), ce qui permettait l'accès direct à l'information.

Au début des années 80, la Banque s'est équipée d'un ordinateur IBM 4331 d'une puissance inférieure à 1 MIPS [12].

A cette période, sous l'impulsion du CIG (Centre Informatique de la Générale), la Banque a commencé la mise au point et l'utilisation d'un environnement de programmation appelé "ICE" [Cfr III].

A cette époque a démarré la saisie décentralisée du service du Change sur TEXAS-INSTRUMENTS.

L'année 1982 a vu le passage à un IBM 4341 d'une puissance de 1,2 MIPS.

En 1983, la saisie décentralisée des Opérations Diverses sur TEXAS-INSTRUMENTS a été mise en place. Cette année fut aussi marquée par l'installation dans un service, en l'occurrence celui des Opérations Diverses, des premiers terminaux IBM, ce qui permettait l'accès à l'information de façon interactive.

Les années 1985 et 1986 ont été marquées par la disparition complète des cartes perforées.

Les années 1986 et 1987 ont vu le passage à un ordinateur central (mainframe) IBM 4381 d'une puissance de 3,5 MIPS. C'est à cette époque également que la micro-informatique a démarré et que des PC ont été installés dans la Banque.

En 1989, l'IBM 4381 disposait d'une puissance de 6 MIPS. Cette même année, on recensait la présence de plus de 220 terminaux IBM dans les différents services de la banque.

En résumé, les dates importantes de l'évolution informatique sont :

1968	IBM 360/20 : enregistrement des opérations et comptabilité.
1970	Remplacement de l'unité centrale.
1978	IBM 370/115 muni d'un DOS.
1980	IBM 4331 (1 MIPS). Saisie décentralisée du change sur TEXAS-INSTRUMENTS. Mise au point et utilisation de ICE.
1982	IBM 4341 (1,2 MIPS).
1983	Saisie décentralisée des Opérations Diverses sur TEXAS. Premiers terminaux IBM dans les services.
1985	Disparition complète des cartes perforées.
1986	IBM 4381 (3,5 MIPS). Premiers PC.
1989	IBM 4381 (6 MIPS, 220 terminaux).
1990	....

Fig. II - 1

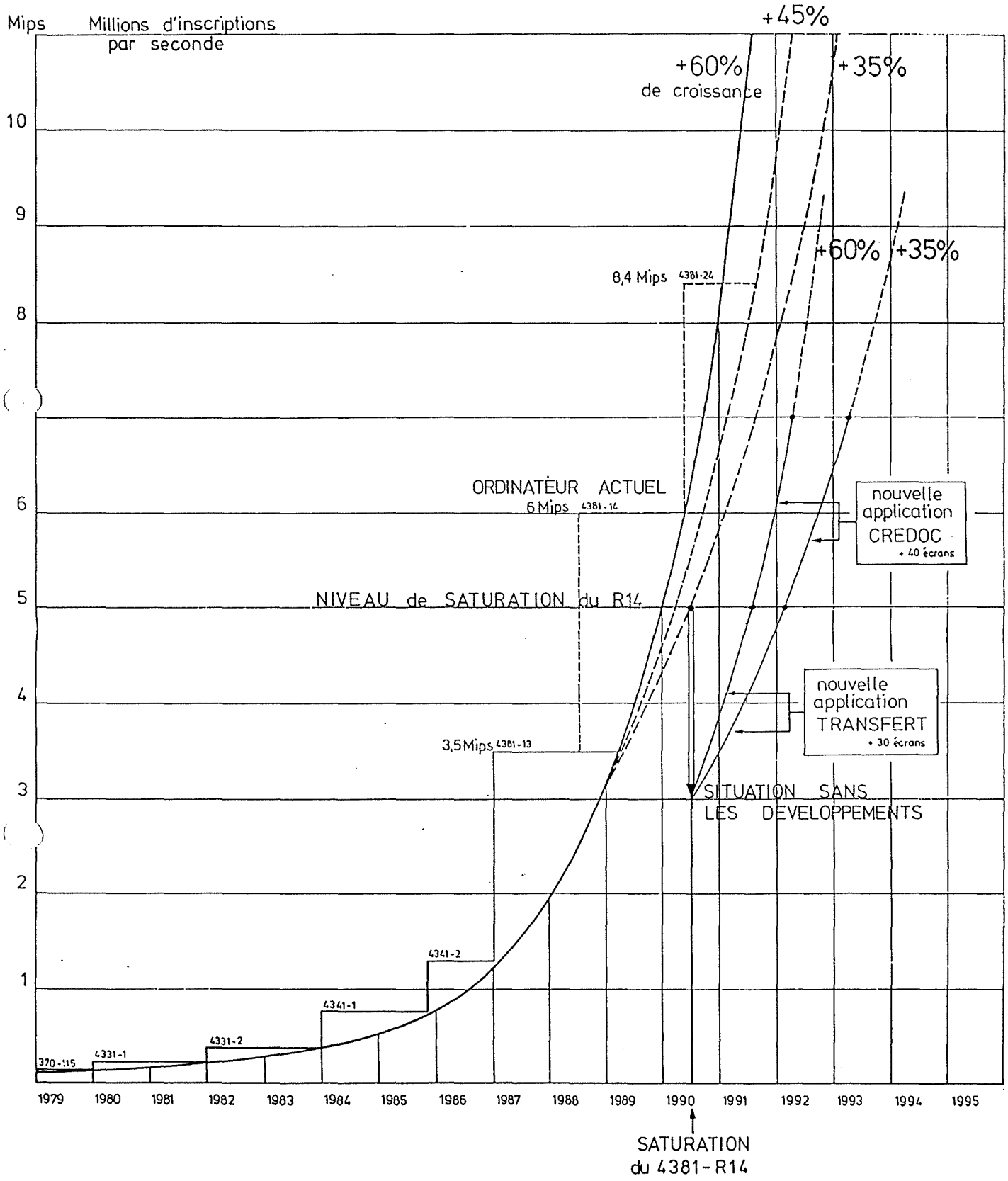
La Banque est aujourd'hui parvenue à un point de quasi saturation de son système informatique. Avec une croissance annuelle de la puissance calcul approchant les 35 %, le mainframe IBM 4381 a presque atteint sa configuration maximale [Cfr page suivante, Fig. II - 2]. Il faut dès lors penser à son renouvellement.

La Belgolaise étudie actuellement l'acquisition soit d'un second 4381, soit le passage dans la gamme supérieure du constructeur, soit enfin, de se tourner vers la concurrence. Mais toutes ces possibilités posent des problèmes.

1) La solution du dédoublement du système actuel, soulève différents problèmes cruciaux :

   Qui dit deux machines, dit deux fois plus d'espace. Or l'espace est, lui aussi, une ressource rare et limitée. Et il semble très difficile de libérer les locaux nécessaires.

- Fig. II.2. -


Mips
Millions d'inscriptions par seconde
+45%
+60% de croissance
+35%
+60%
+35%
8,4 Mips 4381-24
ORDINATEUR ACTUEL
6 Mips 4381-14
nouvelle application CREDOC
+ 40 écrans
NIVEAU de SATURATION du R14
nouvelle application TRANSFERT
+ 30 écrans
3,5 Mips 4381-13
SITUATION SANS LES DEVELOPPEMENTS
4341-2
4341-1
4331-2
4331-1
370-115
10
9
8
6
5
4
3
2
1
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
SATURATION du 4381-R14

De plus, cette première solution n'est pas des plus économiques car elle implique le dédoublement des frais de location des logiciels de base (Système d'exploitation VM-VSE d'IBM).

Par contre, cette solution présente d'indéniables avantages : elle permettrait en effet de séparer physiquement les unités de développement et de production et d'assurer dès lors un "backup" aisé de cette dernière unité.

2) La deuxième solution, celle du passage à la gamme 3090 d'IBM, est elle aussi problématique.

Au-delà de l'investissement important qu'elle nécessite, cette solution soulève à nouveau un problème d'infrastructure. En effet, contrairement à la gamme moyenne, dont le refroidissement est assuré par ventilation forcée, les unités de la gamme 3090 sont équipées d'un système à refroidissement liquide qui nécessite une surface et des installations beaucoup plus importantes.

De plus, cette évolution pour être réellement intéressante impliquerait l'utilisation d'un OS [13] plus puissant, plus complexe et plus coûteux (MVS [24]). Il faudrait aussi réécrire certaines parties de l'environnement ICE pour qu'il tire profit de ce nouveau système d'exploitation. Cela a comme conséquence immédiate de ralentir le développement de nouvelles applications et de gonfler pour longtemps le budget de la location de l'OS.

Enfin, si le passage à la gamme supérieure répond au problème de saturation de la configuration centrale, il n'apporte aucun bénéfice pour les utilisateurs.

Ces contraintes de budget, d'infrastructure et de maintenance semblent condamner cette deuxième possibilité.

3) La dernière possibilité, le passage à la concurrence par l'achat de matériels compatibles, est elle aussi problématique :

Le nouveau matériel, sera-t-il encore entièrement compatible avec l'OS et les multiples applications existantes ? Si cette compatibilité n'était pas réalisée à 100 %, les efforts d'adaptation des logiciels de base risqueraient d'êtres énormes.

Signalons cependant qu'une autre société fait fonctionner l'environnement ICE [Cfr III] sur du matériel Hitachi sans aucun problème. Ce problème de compatibilité semble donc être actuellement résolu.

D'autre part, IBM, restant fournisseur de l'OS continuerait-il à assurer la maintenance avec autant de zèle qu'auparavant... ? On le voit, si la compatibilité est un problème actuellement résolu, le service après vente, la maintenance et le suivi de la gamme en sont d'autres bien plus problématiques à long terme.

Bien que cette solution d'achat de matériels compatibles semble la plus écomomique au premier abord, les coûts et risques secondaires qui lui sont liés, sont tels qu'ils hypothèquent fortement cette dernière éventualité.

1.2. TACHES : Les activités et les services informatisés

De nombreuses applications fonctionnent à l'heure actuelle sur le système informatique de la Belgolaise. Nous en proposons ci-dessous la liste exhaustive. Certaines sont de type "BATCH", notées "BT", et d'autres sont de type interactif ou encore temps réel, notées "TP". Enfin, il y en a de nouvelles, notées "NV" et d'autres plus anciennes "AC".

BZA	Signalétique clients et banquiers	TP		NV
BZB	Gestion des crédits	TP	BT	AC
BZC	Paiement étranger	TP	BT	AC
BZD	Portefeuille-Créd. Documentaires	TP	BT	AC
BZE	Dealing - Trésorerie	TP	BT	NV
BZG	Gestion des dépôts à terme	TP	BT	AC
BZH	Titres - gestion fortune	TP	BT	AC
BZK	Paiements BEF-Opérations Diverses	TP		AC
BZL	Interface Belgolaise-Londres		BT	AC
BZM	Messagerie	TP		NV
BZP	Gestion personnel-Paie	TP	BT	NV
BZS	Comptabilité Clients et Centrale	TP	BT	NV
BZT	Electronic banking	TP		NV
BZU	Saisie universelle	TP		NV
BZV	Visa-Service Client-cartes crédit	TP		NV
BZZ	Téléphonie	TP		NV

Fig. II - 3

L'ensemble des applications représente 2.300 programmes, tandis que les applications "TP" représentent 1.100 programmes.

La tendance la plus nette qui se détache du tableau précédent est un remplacement progressif des applications différées (BT) par des applications en temps réel (TP).

Mais il existe aussi des domaines non encore couverts par une application informatique tels que :

- Crédits Internationaux.
- Comptabilité Analytique.

1.3. STRUCTURES : L'organisation du département

Le département informatique est divisé en trois services séparés : l'organisation, le développement et l'exploitation.

1) Le service organisation dont les missions sont les suivantes :

- l'analyse primaire des besoins,
- la définition des objectifs et des limites d'un projet,
- la formation aux fonctions bancaires,
- la coordination inter et intra services,
- la gestion des applications bureautiques,
- la gestion de la documentation bancaire,
- la gestion des implantations,
- ...

2) Le service développement dont les missions sont les suivantes :

- la cohérence de l'architecture logicielle et des données,
- l'analyse/programmation des applications,
- la formation aux fonctions informatiques,
- ...

3) Le servive exploitation lui-même subdivisé en cinq Fonctions distinctes :

- La "Fonction SAP" est composée de quatre personnes et de quatre opérateurs qui sont responsables du support des applications et de la préparation des traitements batch. Ils travaillent en équipe, l'une de 6 à 14 heures et l'autre de 14 à 23 heures.

- La "Fonction STR". Composée de deux personnes, elle réalise le "support temps réel" des applications. Elle est la première en contact avec les utilisateurs finaux du mainframe.

- La "Fonction SYSTEME", composée de deux ingénieurs système, assure la maintenance et le fonctionnement des l'OS du mainframe et du noyau ICE.

- La "Fonction SAISIE ENCODAGE CONTROLE", qui compta jadis jusqu'à 20 personnes, et qui n'en compte plus aujourd'hui que 4, subsiste encore pour quelque temps mais est clairement en voie de disparition. Cela a été possible grâce à la réalisation d'un outil de saisie universelle. Toute opération comptable est saisie dorénavant via cette application de façon interactive et décentralisée. Auparavant, les différents services rédigeaient des bordereaux qui étaient saisis sur le site central par le biais d'encodeuses. Aujourd'hui cette étape tend à disparaître. Les agents utilisent maintenant les facilités de l'outil de saisie universelle des opérations comptables depuis leurs terminaux installés dans les différents services. Cependant, le contrôle et la cohérence des données seront toujours assurés par la cette fonction.

- La "Fonction SUPPORT AFRIQUE" est assurée par chacune des autres Fonctions. Les filiales africaines connaissant d'énormes difficultés pour joindre leur correspondant informatique : constructeurs, dealers, etc... La Belgolaise assure donc pour eux les fonctions d'intermédiaire en Europe.

Cependant, les compétences des trois services organisation, développement et exploitation ne sont pas toujours nettement distinctes et, par exemple, il peut arriver que le développement empiète sur les tâches de l'exploitation ou de l'organisation.

1.4. INDIVIDUS : Evolution du personnel dans le département

Si l'informatique est un outil de rationalisation pour l'organisation, le service lui-même n'échappe pas à sa propre rationalisation qui se marque moins par une diminution de la quantité globale d'emploi, que par une modification de la répartition du personnel entre les différentes fonctions du service. On constate une forte diminution de la fonction exploitation au profit des fonctions développement et organisation.

Par exemple, les efforts d'automatisation ont permis de diminuer le personnel de saisie et de manutention (bandes, listing...). Parallèlement, la partie analyse-programmation a doublé son effectif en 8 ans, passant de 9 à 17 personnes aujourd'hui [Cfr page suivante, Fig. II - 4]. Ce phénomène est explicable par le nombre important d'applications en chantier, induisant un accroissement important du cadre de conception.

07.02.90

Evolution des effectifs

ORGANISATION - INFORMATIQUE

	1982	1983	1984	1985	1986	1987	1988	1989
DIRECTION-SECRETARIAT	4	5	5	3	3	4	4	4
ORGANISATION	6	4	5	8	9	11	11	8
ORGANISATION AFRIQUE	1	1	1	1	1	1	1	1
ANALYSE-PROGRAMMATION	9	8	8	9	11	11	12	17
CENTRE INFORMATIQUE	24	24	25	24	21	23	21	16
DATA SECURITY OFFICER	0	0	0	0	0	1	1	1
FORMATION	0	0	0	0	1	1	1	1
SOUS-TOTAL	44	42	44	45	46	52	51	48
CONTRATS A DUREE DETERMINEE	0	0	0	0	3	7	8	1
CONTRATS ONEM	0	0	0	0	3	4	3	4
SOUS-TOTAL	0	0	0	0	6	11	11	5
TELECOM	13	13	13	13	13	13	13	13
EXPEDITION	10	10	10	10	10	10	9	8
SOUS-TOTAL	23	23	23	23	23	23	22	21
TOTAL GENERAL	67	65	67	68	75	86	84	74

- Fig. II. 4. -

Enfin un phénomène particulier mérite d'être souligné : l'utilisation, durant les "périodes de coups de feu", d'informaticiens sous contrat à durée déterminée et de stagiaires ONEM. Entre 1986 et 1990, en moyenne, 7 personnes transitaient chaque année par le service.

1.5. TECHNIQUES : L'architecture

a. L'architecture matérielle

Au coeur du système informatique de la Belgolaise, repose un mainframe IBM 4381 d'une puissance actuelle de 6 MIPS. C'est l'une des unités la plus puissante de la gamme 43 d'IBM. Seul, le modèle 24 de cette même gamme possède une puissance supérieure (8 MIPS).

De toute la Banque, il est possible d'accéder à ce système central via plus de 220 terminaux IBM.

A ce mainframe, sont connectés des périphériques classiques et indispensables : unités à disques, unités à bandes, et imprimantes (Cfr page suivante, Fig. II - 5).

L'ordinateur central est connecté :

1) à la Succursale d'Anvers (via une ligne louée RTT),

2) à la Bourse de Bruxelles (via un PC frontal),

3) au réseau SWIFT (via un frontal ST200 Unisys),

4) au réseau TELEX (via un PC frontal),

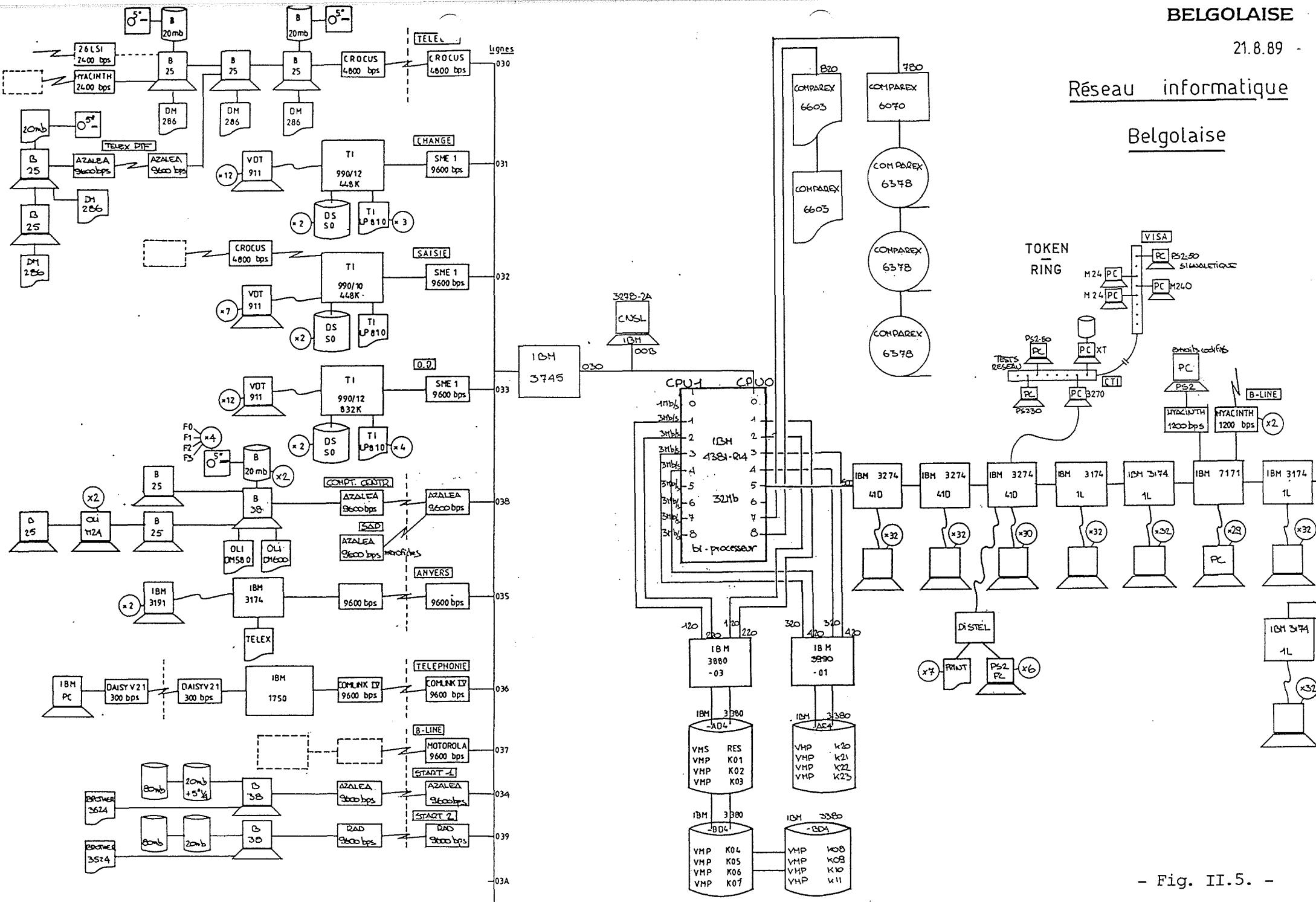
5) à différents PC (via un réseau local de type "TOKEN RING" [15],

6) à différents clients importants au travers des services du produit "B-LINE" qui permet à un client de consulter sa situation à partir d'un terminal installé chez lui.

7) aux trois ordinateurs TEXAS, eux-mêmes connectés à quelques terminaux qui réalisent la saisie décentralisée du service du Change. Le mainframe va lire les données saisies par l'intermédiaire de job BATCH lancé à intervalles réguliers.

Réseau informatique

Belgolaise



- Fig. II.5. -

En règle générale, et pour des raisons de sécurité, le mainframe n'est jamais connecté directement à un autre ordinateur [Cfr Fig. II - 6 et 7 ci-dessous].

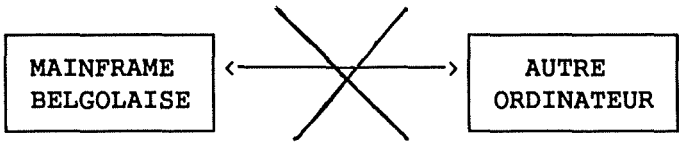


Fig. II - 6

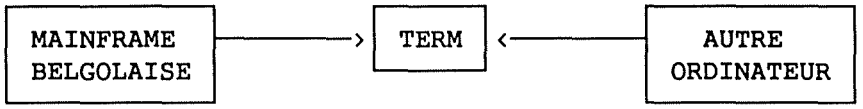


Fig. II - 7

Par exemple, l'ordinateur de la Bourse considère le PC Belgolaise comme une imprimante sur laquelle il écrit. Les caractères envoyés sont stockés momentanément sur le disque dur du PC. Le mainframe peut alors lire les informations de ce disque [Cfr ci-dessous, Fig. II - 8].

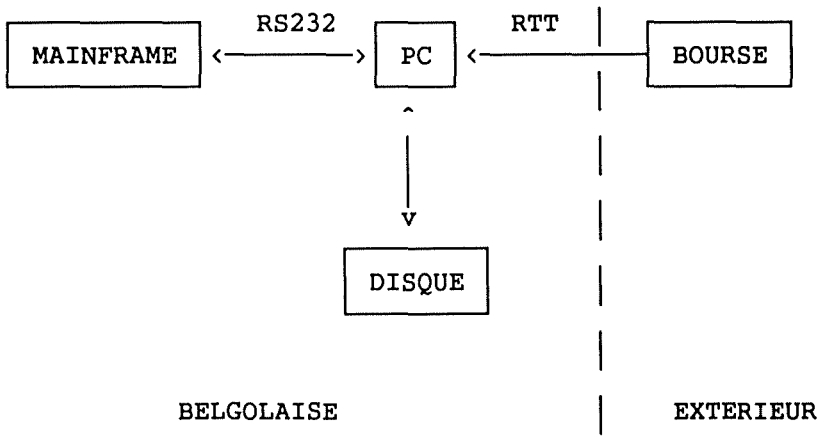


Fig. II - 8

Tout se passe comme si, le PC Belgolaise était un terminal de la BOURSE. De même, le ST200 Unisys est considéré comme un terminal du réseau SWIFT. L'IBM Belgolaise considère aussi ces machines (PC Belgolaise ou ST200) comme des terminaux sur lesquels il lit des données. Il n'est donc pas possible d'exécuter un programmme du système de SWIFT à partir du mainframe de la Banque et vice-versa. Enfin, les terminaux de la Succursale d'Anvers sont pour le mainframe des terminaux comme ceux de la Banque. Il n'y a donc jamais de connexion directe avec un autre ordinateur [Cfr ci-après, Fig. II - 9].

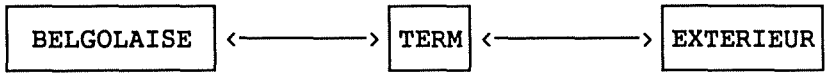


Fig. I - 9

Enfin, dans les différents services sont répartis une cinquantaine de PC IBM et OLIVETTI qui servent principalement aux diverses activités bureautiques : traitement de textes, tableurs, etc.

b. L'architecture logicielle

Examinons l'architecture informatique sous l'angle des logiciels. Le mainframe tourne sous le système d'exploitation VM (Virtual Machine) d'IBM. Cet OS possède la caractéristique de permettre de diviser la machine physique en une multitude de machines virtuelles. Ceci présente l'avantage incontestable de séparer "physiquement" diverses applications. Si l'une des machines virtuelles venait à se bloquer, les autres n'en seraient pas pour autant affectées. Seul le cas où l'unité centrale connaîtrait une panne matérielle affecterait l'ensemble des machines virtuelles.

Dans sa configuration actuelle, il existe cinq machines virtuelles. Chacune d'elles fonctionne à l'aide de son propre système d'exploitation, le VSE d'IBM :

1) La première machine virtuelle assure les fonctions de production des traitements BATCH.

2) La deuxième machine virtuelle, dite "TP1", assure les traitements interactifs des services Visa et Télécom.

3) La troisième machine virtuelle, dite "TP2", assure les autres traitements interactifs et gère les 150 terminaux des différents services répartis dans toute la Banque.

4) La quatrième machine virtuelle assure les fonctions de communication avec les clients extérieurs. La Belgolaise leur offre principalement deux grands types de service :

A) Ceux du service "B-LINE" qui permettent la "consultation en temps réel" des comptes depuis un terminal placé chez eux.

B) Un second ensemble de services permet la délivrance d'extraits de compte. Ceux-ci sont transmis, par l'intermédiaire du réseau commuté RTT, dans le format d'un fichier PC aux normes ABB. On dénombre actuellement une dizaine de clients importants qui sont de cette façon connectés à la Banque. Cette activité connaît en ce moment un fort taux de croissance.

5) Il existe aussi une dernière machine virtuelle qui est exclusivement réservée aux activités de tests des nouvelles applications.

Enfin, il est possible de créer pour chaque programmeur une machine virtuelle mono-utilisateur. Ces petites machines virtuelles assimilables à un PC, fonctionnent sous CMS et servent principalement à l'édition des applications qui sont compilées et testées dans la cinquième machine virtuelle, celle réservée aux tests.

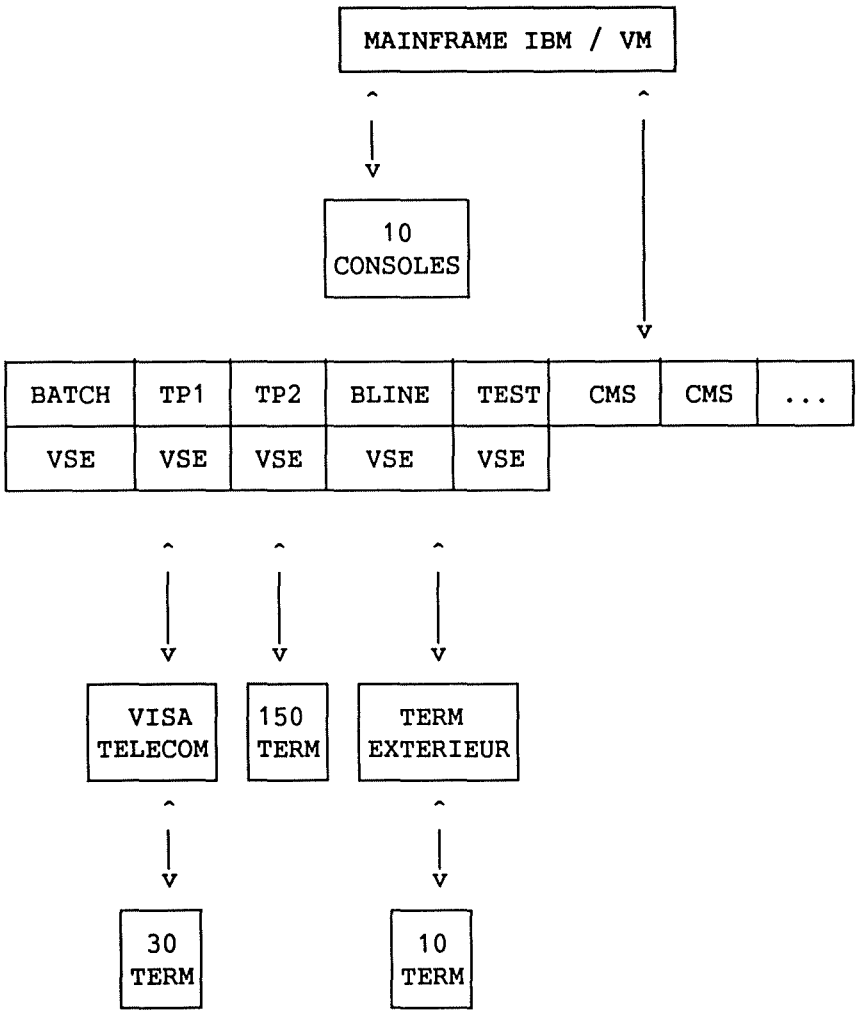


Fig. II - 10

c. L'aspect traitement différé

L'informatique à la Banque est fortement standardisée. Qu'il s'agisse de programmation par lots ou interactive, les méthodes de développement doivent être scrupuleusement respectées.

Ce mémoire s'attachera à décrire avec précision les méthodologies appliquées au développement de logiciels transactionnels interactifs. Nous aurons donc l'occasion d'y revenir. Nous proposons cependant une brève description des méthodes de développement des programmes "batch" parce que ces dernières nous semblent suffisamment originales pour être soulignées.

En règle générale, un programme "BATCH" fait trois choses :

1) Il parcourt des fichiers en réalisant des mises à jour.
2) Il produit des listes.
3) Il trie, formate et envoye ces listes à des destinataires.

La méthodologie BATCH de développement des applications est la suivante. Le programmeur va subdiviser son application en trois modules distincts : "J1", "J2", "J3" [Cfr page suivante, Fig. II - 11].

Le module "J1" a la charge de saisir les différents paramètres du programme. Par exemple, le nom des fichiers sur lesquels il faut travailler.

Le module "J2" est le module qui réalise le parcours et les mises à jour des fichiers. "J2" construit aussi systématiquement un fichier qui contient les différentes listes que l'application doit produire. Ce fichier contient les listes "brutes"; elles ne sont encore ni triées, ni formatées.

Enfin, le module "J3" prend en charge le formatage des différentes listes et l'envoi aux destinataires.

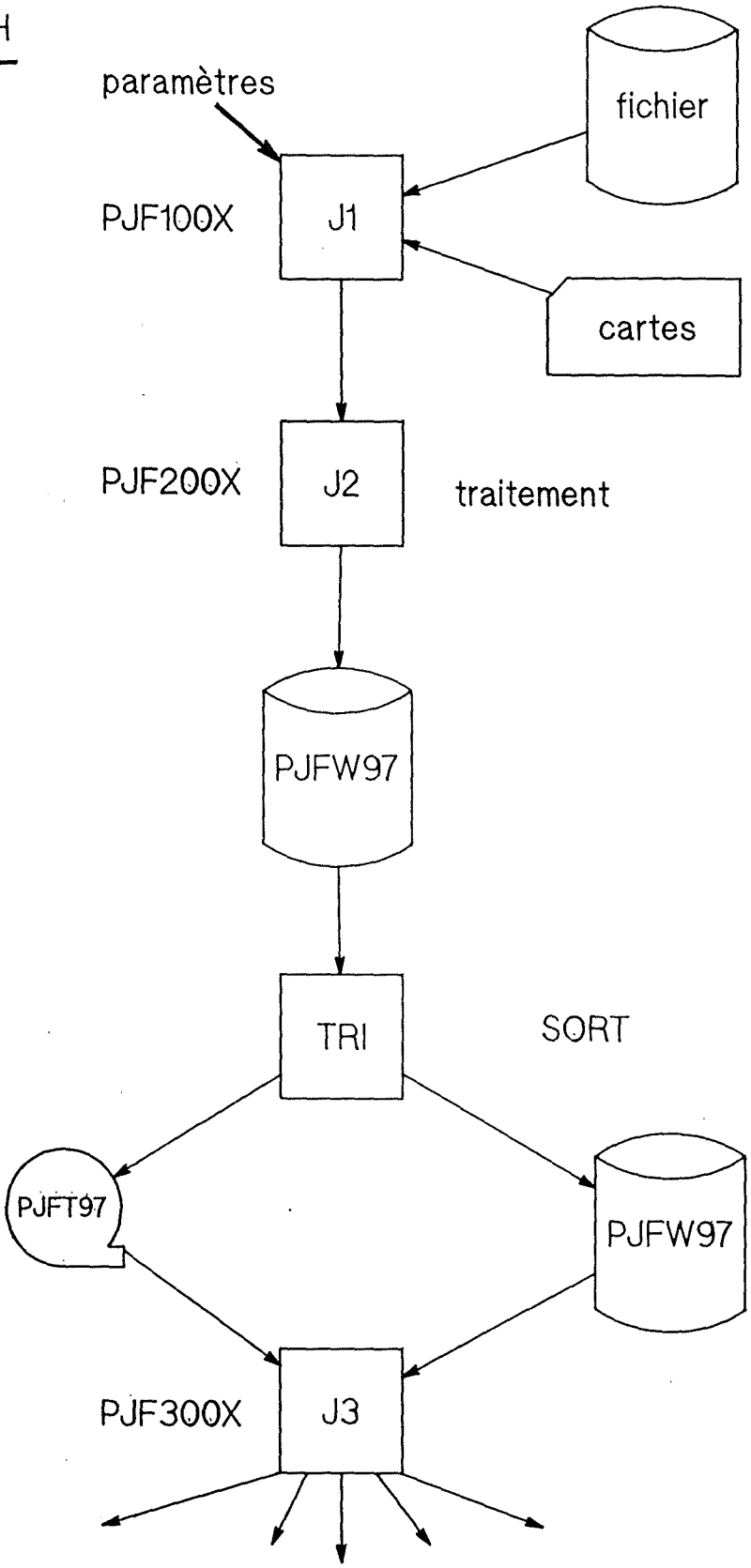
Après avoir écrit différentes applications "BATCH", c'est-à-dire après avoir écrit les différents modules "J1", "J2", "J3", un programmeur peut les enchaîner au sein d'un "PROCESS". De même, différents process peuvent être combinés au sein d'un "JOB".

Un moniteur particulier gère l'enchaînement correct des jobs, des process et des modules.

Les principaux avantages de cette méthode résident premièrement dans le fait qu'il n'y a que très peu de JCL (Job Control Langage) à écrire. Il suffit de demander l'exécution d'un job. Il existe un programme standard réalisant cette instruction et dont le seul paramètre est le numéro du job dont on désire l'exécution.

Ensuite, la standardisation à outrance des programmes "batch" facilite énormément la maintenance, accentue la portabilité des applications, augmente la productivité et la mobilité des programmeurs.

ICE BATCH



- saisie des paramètres
- génération du JCL

32 destinataires en parallèle
- listes
- punchs
- fichiers ICE

Enfin, la séparation physique entre les tâches de formatage et de mise à jour permet de redémarrer le processus en un point quelconque de la chaîne. Cela économise souvent de la puissance calcul, en évitant de reprendre le traitement au début.

Notons pour terminer que le moniteur responsable du séquencement des différents modules "J1" , "J2", ... est actuellement en cours de réécriture. On en profite aussi pour en augmenter fonctionnalité et puissance.

1.6. BUTS : Stratégie informatique

Tâchons maintenant d'examiner quelle est la stratégie poursuivie dans le domaine informatique.

a. Evolution du schéma directeur

De l'avis même d'un responsable, jusqu'aux années 70, la Belgolaise a certainement été en retard sur le plan informatique. Son activité, principalement tournée vers le commerce international, rendait ses opérations des plus complexes à automatiser. Les applications majeures qui furent réalisées jusqu'alors concernaient en premier chef l'activité comptable.

La Banque, équipée d'un IBM de la série 370, investit dans les technologies de l'information et réalisa l'acquisition d'un IBM 4331. Le passage de la technologie à bandes à celle des disques et le remplacement du langage RPG par le COBOL engendrèrent d'importants bouleversements. Il était nécessaire de réécrire l'ensemble des applications à cause des changements d'OS et de langage de programmation.

Sous l'impulsion du Centre Informatique de la Générale (CIG), autre filiale de la Générale de Banque spécialisée dans l'activité de développement logiciel, il fut décidé de faire table rase du passé et de réécrire l'ensemble des applications. Le but de l'opération était de rendre ces nouvelles applications les plus indépendantes possible de la configuration matérielle.

C'est à cette époque que les deux systèmes STI [17] et STICS [18], dédiés respectivement aux traitements batch et transactionnels firent leur apparition. Ce sont, en quelque sorte, les ancêtres de l'environnement ICE [Cfr III].

Cependant, même si, d'un point de vue technique, on pouvait constater d'importants changements, d'un point de vue fonctionnel, les applications étaient peu modifiées.

b. Les fondements du schéma directeur

Le système fonctionna tel quel, jusqu'au début des années 80. A cette époque, il fut décidé une complète réécriture du système dans le cadre de l'environnement de développement ICE [Cfr I - 1.1 et III].

Cette restructuration commença effectivement en 1988 et s'étendit jusqu'à la mi-89. Mais l'ensemble du système ne sera achevé complètement qu'à l'horizon 92.

Ce développement suit trois concepts directeurs :

1) la modularité,
2) l'intégration des applications,
3) l'unicité des données.

- La modularité

L'ensemble du système a été conçu de manière modulaire. C'est-à-dire que les opérations les plus courantes ont été implémentées une fois pour toutes de façon optimale : ce sont les primitives. Et chaque application qui veut effectuer cette opération doit utiliser impérativement la primitive qui la réalise.

AVANT

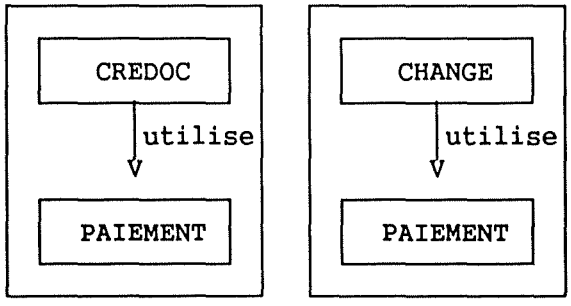


Fig. II - 12

APRES

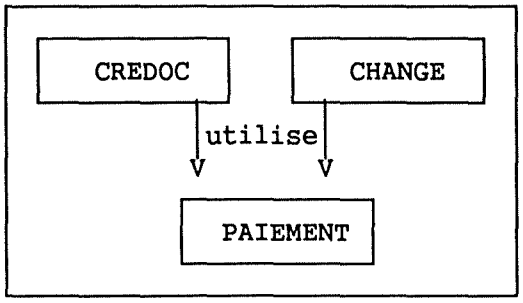


Fig. II - 13

Ces primitives, réalisent souvent des tâches de très bas niveau, mais elles peuvent aussi avoir des buts beaucoup plus sophistiqués, réalisant alors des tâches qui ont un sens pour un banquier, un financier ou encore un comptable [Cfr III - 3.3.].

- L'intégration

Il coexistait précédemment, au coeur de l'organisation, différents systèmes d'information qui parfois s'avéraient redondants ou qui ne savaient pas communiquer entre eux. Le but de l'intégration est de faire disparaître les matériels et logiciels incompatibles et de promouvoir une coopération harmonieuse et efficace entre chaque composant de l'architecture.

- L'unicité des données

Parallèlement, aux problèmes d'intégration des systèmes, il fallait remédier aux incohérences apparues au niveau des données. Par exemple, il existait parfois plusieurs versions du même fichier ou encore un même client pouvait être répertorié sous des adresses différentes, etc. Le principe de l'unicité des données stipule que le système ne mémorise qu'une et une seule fois une information particulière.

C'est notamment dans le but supprimer la redondance induite par les TEXAS-INSTRUMENTS du service Change que ces applications seront, à brève échéance, reconçues afin de respecter le principe d'unicité des données.

c. Conséquences sur l'architecture générale

Ces trois principes guidèrent donc les analystes dans l'élaboration du nouveau système.

Ils les conduisirent à l'élaboration d'une architecture logique du système d'information de la Belgolaise qui se présente dès lors comme suit :

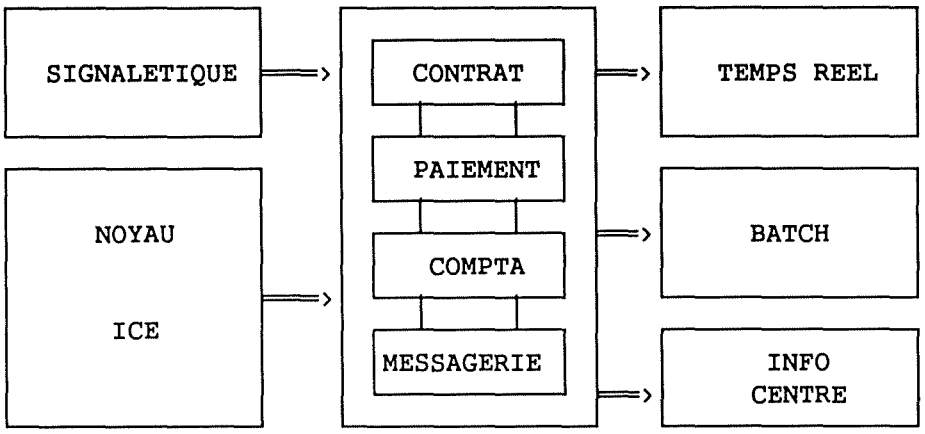


Fig. II - 14

On s'aperçoit que parallèlement aux modules traditionnels de traitement par lots (BATCH) et traitement en temps réel des transactions, une nouvelle famille de modules est progressivement mise en place.

Il s'agit des modules qui assurent des fonctions de :

1) "REPORTING" qui extrait des informations des bases de données pour élaborer des rapports de gestion ou légaux destinés aux autorités;

2) "SECURITE" et "AUDIT TRAIL" qui permettent de préciser qui fait quoi et quand, afin que les transactions critiques soient suivies de près.

   Il ne s'agit pas simplement d'un souci de sécurité mais aussi de fiabilité : "L'ordre de paiement a-t-il été effectué à temps ? Le message Swift a-t-il été bien envoyé ? Ce paiement correspond-il bien à un ordre d'un client ?".

3) STATISTIQUES qui mesure l'activité quantitative de la Banque.

De manière globale, on appréhende cette approche comme un "INFO-CENTRE" pour reprendre le terme commercial courant. Mais à la Belgolaise, l'info-centre prend une forme assez nouvelle. Il s'agit d'un ensemble de programmes tournant sur le mainframe et utilisant ses BD (Bases de données). Capable de réaliser des agrégations des requêtes et d'en mémoriser les résultats successifs à des fins statistiques, cet outil fonctionnera la nuit et durant les périodes creuses.

Les trois concepts fondamentaux n'ont pas seulement servi de phares durant l'élaboration globale, mais aussi durant toute la conception détaillée de chacun des composants. C'est pourquoi, les modules de niveau application (contrat, paiement, messagerie et compte) sont non seulement autonomes les uns des autres, mais chacun utilise aussi les fonctions des autres.

On peut aussi représenter la hiérarchie des différents composants logiciels de cette architecture :

NIVEAU TRANSACTIONNEL - NIVEAU BATCH - NIVEAU REPORT

NIVEAU APPLICATIONS
(contrat, paiement, messagerie compte)

NIVEAU ICE

NIVEAU OS

Fig. II - 15

D'autre part, sur le plan architecture et hiérarchie des applications, la tendance actuelle n'est plus de voir la comptabilité générale comme le coeur du système d'information. Le compte aujourd'hui n'est plus le concept central. La comptabilité générale est maintenant considérée comme une application au même titre que les autres.

Aujourd'hui à la Belgolaise, c'est l'information qui est devenue le concept central.

AVANT

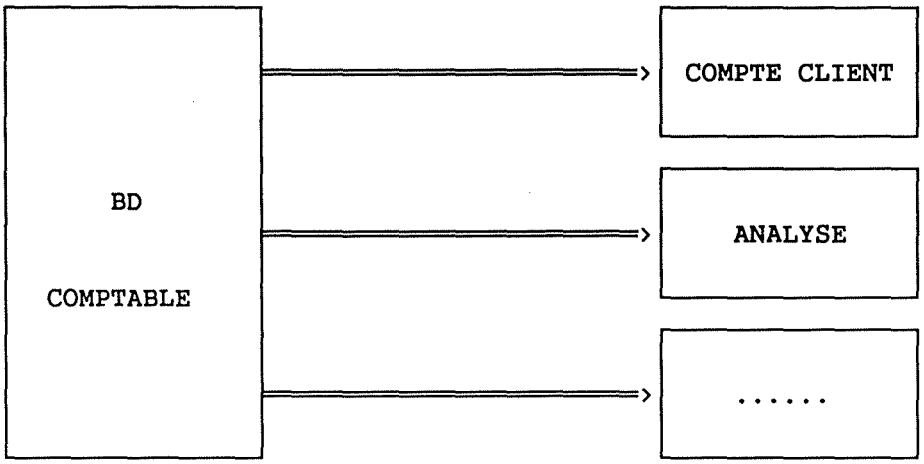


Fig. II - 16

AUJOURD'HUI

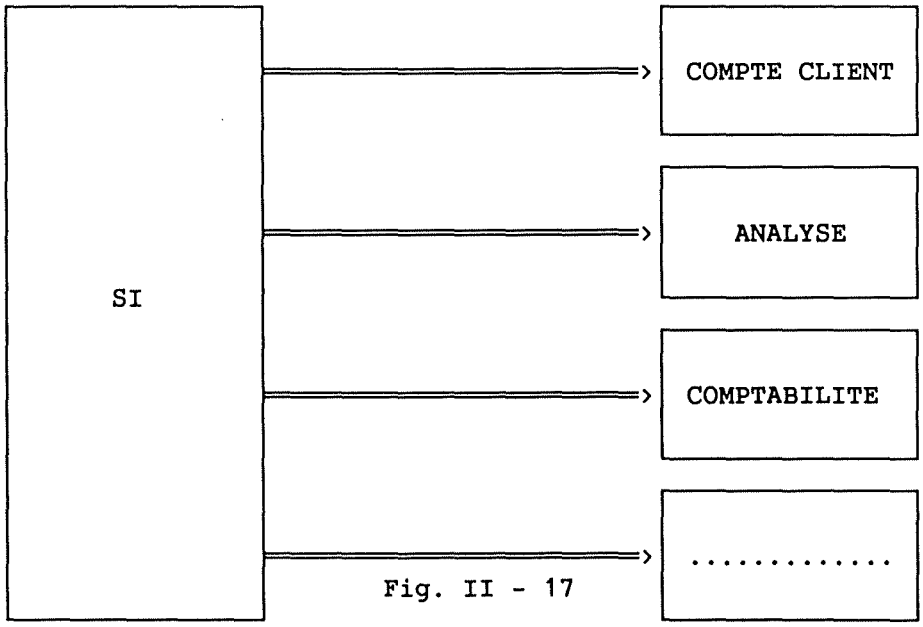


Fig. II - 17

Signalons enfin que la politique de la BELGOLAISE concernant les logiciels est d'acheter les logiciels de base OS, SGBD [16], etc ..., tandis qu'elle préfère développer elle-même les applications spécifiques à son activité commerciale, comme par exemple, la gestion des crédits documentaires.

En résumé, le schéma directeur de la Banque repose sur trois piliers : modularité, intégration et unicité des données. La Belgolaise a, dans un premier temps, développé l'environnement ICE, puis a lancé la réécriture de l'ensemble des applications. Celle-ci démarra durant la seconde partie des années 80 et on peut envisager qu'elle se terminera en 1992.

2. L'APPLICATION DU MODELE DE NOLAN

La section précédente décrit les structures organisationnelles et techniques du système informatique. Cette section propose de replacer ce système dans le cadre du modèle de Nolan [NL79] afin de fournir une perspective diachronique et prospective de la présentation de l'informatique à la Belgolaise.

2.1. La théorie

"Partant de cette vision (évolutionniste) de l'organisation, le rôle de l'informatique apparaît sous un jour nouveau. Celle-ci, du fait de sa puissance de traitement de l'information et sa capacité de mise en mémoire, permet d'augmenter la capacité cognitive globale de l'organisation et de déplacer les frontières de sa rationalité.

Cette image de l'informatique comme élément de la structure de l'organisation et comme support de sa capacité décisionnelle allait connaître une audience exceptionnelle dans le domaine de l'informatique. R. Anthony allait même lui donner une dimension prospective. En effet cet auteur, partant d'une distinction établie par H. Simon entre décision programmée et décision non programmée s'attacha à l'étude des caractéristiques des décisions prises par les différents niveaux hiérarchiques de l'organisation.

Il en dégagea un modèle en trois couches allant des décisions les plus structurées aux décisions les moins programmables dites de stratégie...

C'est sur base de cette image que R. L. Nolan [NL79] allait construire son modèle d'évolution de l'informatique dans l'entreprise" [LB89].

"Nolan analyse le rôle du département informatique dans le sens d'une évolution continue, partant d'un rôle purement instrumental portant essentiellement sur l'exploitation de l'ordinateur et le développement d'applications vers un rôle stratégique de gestionnaire de l'ensemble des ressources informationnelles de l'entreprise" [LB89].

L'auteur du modèle propose donc une analyse diachronique de la fonction informatique. Il émet l'hypothèse que l'ensemble du processus d'informatisation d'une entreprise peut être interprété comme un processus d'apprentissage des technologies de l'information. Il s'agit en réalité d'un quadruple apprentissage imbriqué des différents acteurs concernés : les gestionnaires, les utilisateurs, les informaticiens et les constructeurs. Cet apprentissage organisationnel se marque par une succession d'étapes typiques par lesquelles l'entreprise doit passer.

Ces étapes sont au nombre de 6, à savoir :

1) Initiation,
2) Contagion,
3) Contrôle,
4) Intégration,
5) Gestion de la ressource information,
6) Maturation.

Chacune de ces étapes est identifiable par des valeurs particulières de variables qui sont considérées comme des critères de distinction entre les différentes étapes.

Les variables sont elles aussi au nombre de 6 :

1) Les dépenses réalisées par le centre informatique.
2) Les tâches réalisées par le centre informatique.
3) La technologie utilisée.
4) L'attitude de la direction à l'égard de l'informatique.
5) L'attitude des utilisateurs à l'égard de l'informatique.
6) Le statut des informaticiens.

Le but des sections qui vont suivre est de préciser l'état d'évolution du département informatique de la Belgolaise, en terme de chacune des variables du modèle de Nolan. De cette façon nous pourrons en tirer des conclusions prédictives sur son évolution globale.

2.2. Les dépenses

La première et la plus importante des variables du modèle, ou du moins, celle à qui son auteur accorde la plus grande importance, est sans aucun doute celle des dépenses consenties par l'entreprise dans le domaine informatique. Il la pondère au moyen du chiffre d'affaires de l'année correspondante.

Ne disposant pas de ce dernier renseignement, nous avons pris la liberté de le remplacer par le bénéfice net. Le tableau ci-après, présente la synthèse de ces données.

	84	85	86	87	88	89	90
Invest.	9541	13028	18191	14050	22375	21243	19186
Assist. ext.	774	1009	1261	2043	2091	3515	8500
Frs. fonct.	11453	12575	15357	18206	20590	20607	23950
Divers	56	208	5290	4551	5004	6999	6050
Total dép.	21824	26820	40099	38850	50060	52364	57686
Bénéf. net		421	437	430	434	452	
Dép/Bénéf		63	91	90	115	115	

En millier de francs belges Sources : [BL88]

Fig II - 18

On constate directement sur les deux graphiques qui suivent [Cfr Fig. II - 19 et 20] qu'aussi bien la courbe des dépenses informatiques, que le rapport entre ces dépenses et le bénéfice connaissent de 1984 à 1986 un accroissement rapide, puis se stabilisent durant les deux années suivantes pour enfin repartir plus sagement à partir de 1988.

Bien entendu, cette analyse gagnerait grandement, si on avait pu la réaliser sur les 20 dernières années. Malheureusement, toutes les données nécessaires n'ont pu être rassemblées.

Fig. II.19.

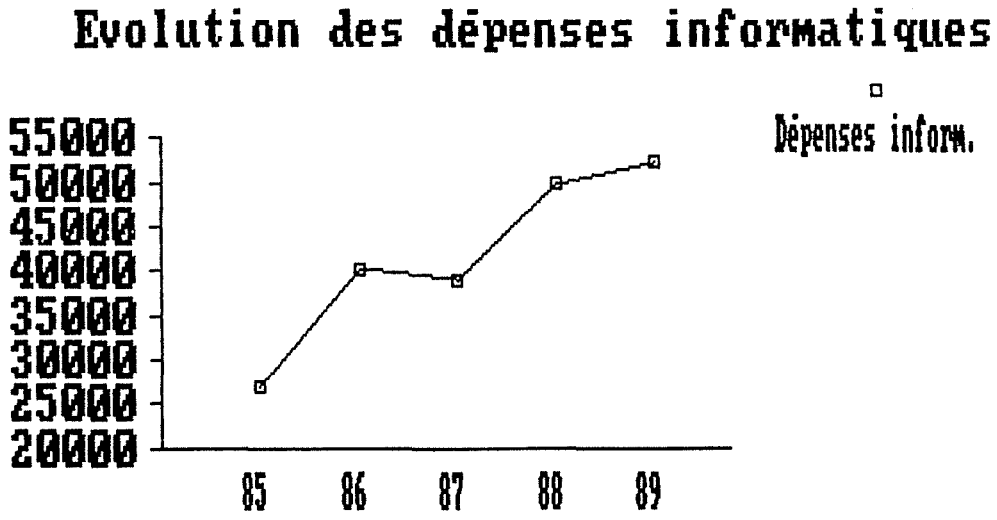
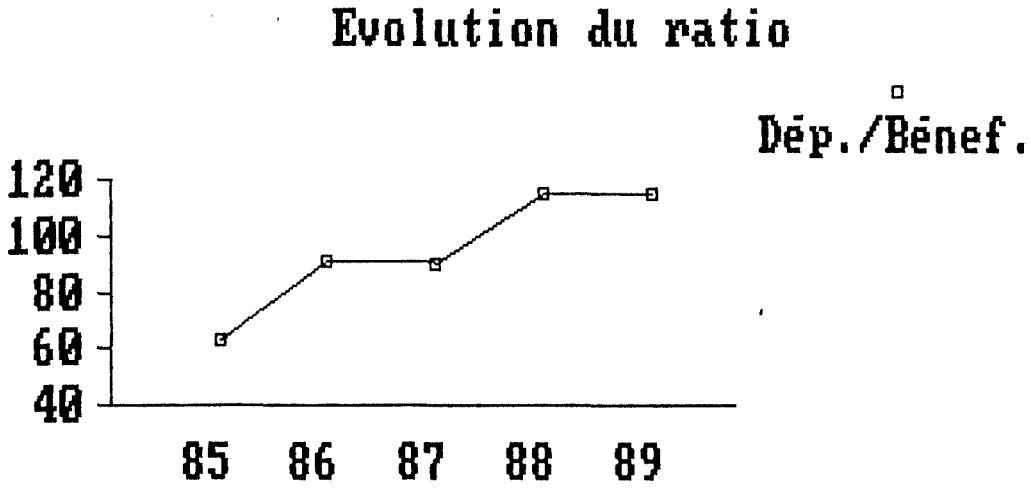


Fig. II.20.



Cependant, le palier que l'on discerne clairement en 1986 et 1987 peut être interprété comme le symptôme d'une transition entre une phase de contagion et une phase d'intégration. Cette transition, Nolan la définit comme étant une phase de contrôle. Durant cette phase, les dépenses informatiques sont en diminution ou en stagnation par rapport au chiffre d'affaires. Ce ralentissement peut être interprété, selon l'auteur du modèle, comme une réaction des gestionnnaires face à l'hémorragie financière que représentent les charges des systèmes d'information lors de leur développement. Avant et après cette période, le management consent à accroître les dépenses dans ce domaine.

Parallèlement, dans son dernier rapport [AB89], l'Association Belge des Banques (ABB [14]) donne des ratios ou des montants d'investissements informatiques rapportés au nombre d'agents.

On constate que 46 % des investissements faits par les banques sont consacrés à l'achat de matériel informatique. A la Belgolaise, il s'élève à plus de 58 %.

Mais cette différence s'explique. C'est le rapport entre les investissements en informatique et les autres investissements. Or, nous l'avons montré, la Banque ne possède qu'une succursale et aucune agence. Dès lors, elle ne réalise que très peu d'investissements immobiliers.

En revanche, le même rapport [AB89] nous apprend que l'achat de matériel informatique représente par employé une dépense annuelle moyenne de 115.000 Frs. A la Belgolaise, il n'est, si l'on peut dire, que de 45.696 Frs par agent (soit 39 %). Si l'on tient compte de la maintenance et du développement les chiffres passent à 334.000 Frs pour l'ABB et à 156.352 Frs à la Belgolaise (soit 47 %).

Une première conclusion pourrait être qu'à la Belgolaise, l'activité logicielle est, soit moins intense, soit coûte moins cher que dans le reste des banques belges.

Cependant, une autre explication peut être ébauchée. Nous savons que la Belgolaise ne possède aucune agence. Elle ne doit donc pas réaliser de lourds investissements en réseau de télécommunication. C'est probablement cette caractéristique qui fait que son ratio est si faible.

En conclusion de cette section, nous affirmerons que la Belgolaise quitte aujourd'hui l'étape de contrôle et s'engage dans l'étape d'intégration.

2.3. La technologie

Quelle a été l'évolution de la technologie à la Banque et où se situe-t-elle à l'heure actuelle ? Pour plus de détails nous invitons le lecteur à se reporter à la section 1.1. du chapitre II consacrée à l'évolution du système d'information. Nous nous contenterons ici d'en reprendre les grandes lignes et d'en tirer quelques conclusions.

Si en 1968 à Paris, on parle d'une révolution, à la même époque à la Belgolaise, c'est de la révolution informatique dont il est question. Cette dernière est principalement placée sous l'égide du traitement différé ou "Batch". Les cartes perforées tiennent alors le haut du pavé.

La première partie de l'année 79 connaît, elle aussi, sa révolution. Il s'agit de l'apparition de la saisie décentralisée sur Texas-Instrument. Les services du Change et des Opérations Diverses "descendent" sur ces ordinateurs départementaux. Les cartes perforées, quant à elles sont à jamais reléguées aux oubliettes.

Enfin, la troisième mais certainement pas la dernière évolution technologique, le réseau local d'IBM, le "Token Ring" [15], fait son apparition dans l'immeuble du Cantersteen. Il permet aux différents ordinateurs d'échanger entre eux les données qu'ils contiennent. L'IBM central et quelques PC sont reliés de la sorte.

Ces différentes révolutions suivent exactement l'évolution prévue par Nolan. Du traitement différé à l'utilisation d'un réseau, en passant par l'utilisation de SGBD [16] et des ordinateurs personnels à des fins bureautiques, la Belgolaise suit pratiquement chacune des étapes du modèle : Initiation, contagion, contrôle...

Aujourd'hui elle se trouve être en pleine transition : plus vraiment en phase de contrôle, pas encore tout à fait en intégration. L'organisation fait encore l'apprentissage des nouvelles technologies informatiques. Preuve en est, si cela était encore nécessaire, le présent travail de développement, d'évaluation et de compréhension des techniques de conceptions de logiciels sur un réseau de PC.

2.4. Les tâches

Anthony, dans son modèle, montre que toute organisation peut être hiérarchisée en trois niveaux d'activités. Il s'agit des activités opérationnelles, de gestion et stratégiques.

Nolan, pour sa part, émet l'hypothèse que tout système d'information durant son évolution passe par les six étapes de son modèle. De ces six phases, il est possible de détermimer trois mouvements de base à savoir [LB86] :

1) le support des activités opérationnelles,
2) le support des activités de gestion,
3) le support des activités stratégiques.

L'examen attentif de ces deux modèles met clairement en évidence le fait que le domaine d'application des systèmes d'information poursuit un mouvement ascendant dans la hiérarchie des activités de l'organisation : depuis les activités élémentaires vers les plus stratégiques.

Qu'en est-il à la Belgolaise ? Nous allons, dans cette section, essayer de démontrer que ces hypothèses sont également vérifiées pour cette entreprise.

En 1968, la première des fonctions qui a été confiée à l'ordinateur a été l'automatisation de la tenue des comptes clients.

La comptabilité est dans le modèle de Nolan prototypiquement le genre d'opérations que l'on tente d'informatiser. De plus, la mise à jour quotidienne du solde des comptes représente une charge opérationnelle très importante. C'est la période d'initiation.

La plupart des sociétés tentent ensuite de réaliser ou d'assister leur production à l'aide d'un système d'information. Pour une banque, il s'agit par exemple de l'implémentation de logiciels de gestion des crédits. Ce fut là l'une des premières applications de type opérationnel qu'a développée le service informatique. Citons de même, l'automatisation du traitement des virements, chèques et autres domiciliations. Nous sommes ici, clairement, en présence de l'étape de contagion.

Le lancement du projet ICE (Cfr III), et la réécriture progressive de l'ensemble des applications est parfaitement symptomatique de la phase de contrôle vue sous l'angle des tâches confiées à l'informatique. La direction de ce service, soucieuse de maîtriser les coûts de développement et de maintenance, lance une grande offensive de standardisation. Nous montrerons plus loin, en quoi ICE, sa méthode et son langage constituent résolument un outil de standardisation.

Aujourd'hui, presque 90 % des applications fonctionnent dans l'environnement ICE. L'objectif des analystes est de reconcevoir les plus anciennes applications pour les améliorer et maximiser autant que possible la valeur ajoutée de chacune.

Notons aussi qu'aujourd'hui, un des principaux soucis des analystes est de produire de nouveaux états imprimés qui ne seraient plus utiles pour les opérations courantes mais bien pour les activités de gestion de l'organisation.

En revanche, on ne connaît pas encore d'applications informatiques majeures qui assisteraient significativement dans son travail la direction générale. Nous pensons notamment aux SIAD [13] qui fonctionnnent dans d'autres entreprises. Cependant, plusieurs applications de ce type sont en ce moment à l'étude et devraient voir le jour d'ici un an, à savoir : la production de rapports, d'audits et de statistiques.

En conclusion, comme à la section précédente : la Belgolaise se trouve aujourd'hui en phase de transition, entre la phase contrôle et celle d'intégration.

2.5. L'attitude de la direction

La direction est parfaitement consciente de l'importance des technologies de l'information. D'ailleurs, elle a déjà consenti d'importants efforts financiers dont le but était l'augmentation de la productivité des développeurs et l'amélioration de la qualité des services informatiques rendus à la clientèle.

Il semble que les gestionnaires de l'entreprise soient d'accord de maintenir constant le niveau des investissements de développement. Par contre, ils seraient plus réticents à réaliser de nouveaux gros investissements. Ils estiment que, du fait de l'activité spécifique de la Belgolaise, l'informatisation ne puisse lui apporter un avantage concurrentiel décisif. Elle estime en effet que la Belgolaise n'est pas une banque grand public et donc n'est pas soumise à une concurrence de type "produit" mais plutôt de type "service". Elle doit donc focaliser ses efforts sur la qualité et la disponibilité de son information.

Il est intéressant de confronter cette dernière remarque avec les recommandations de Michael J. Earl :

"...At Oxford in the last year, we have had two companies seek help in information management education because they have been disadvantaged by rivals using IT (information technology) competitively. A third company with whom we have working now realizes that it is not achieving its IT plans quickly enough as both its rivals and custumers embrace IT boldly and rapidly.

Indeed we find increasingly that executives who are close to competitive force can quote very quickly examples of suppliers, rivals, customers, or new entrants who are making IT strikes. Commonly these executives do not realize how much time and effort will be required either to react or to make similar moves.
Indeed this is one of the attractive feature of IT as competitive weapon. It cannot always be copied quickly and easily.

... In other words in sectors where business is either being redefined by IT or is dependent on IT, poor systems can be a competitive disavantage and impair survival..."
[ER88].

Cependant il constate aussi que :

"... IT capability and creativity may offer potential competitive advantage, but they are not sufficient in themself if sustainable competitive advantage is the aim...

... Non IT factors must also be analysed when considering use of IT for competitive advantage..." [ER88].

Il faut noter que, parallèlement, la Belgolaise est prête à investir dans son infrastructure de télécommunication, malgré que celle-ci soit déjà excellente. Rappelons qu'elle est la seule entreprise belge à disposer d'une liaison téléphonique permanente par satellite avec Kinshasa.

Ce fait est particulièrement intéressant puisque les télécommunications sont le support indispensable pour la circulation des informations.

D'autre part, nous avons déjà souligné le caractère critique des liaisons avec les clients résidant à l'étranger. C'est dans cet esprit que la Banque a déjà développé la gamme de ses produits "B-LINE" [23] (extraits de comptes électroniques et cash management).

Les télécommunications sont donc un facteur on ne peut plus critique de succès pour l'ensemble des services et dès lors l'entreprise. C'est aussi l'avis de David A. Runge et Michael J. Earl :

"...According to Bell [BL79], telecommunications forms the central infrastructure of an information society in which information becomes the strategic resource. For businessmen, the economic implication of telecommunications are becoming ever clearer through the 1980's; telecommunications eliminates barriers of geography and time on service and co-ordination [KN86]. Telecommunications networks provide the information highways over which new products and services

can be offered, thereby redefining concepts of custumer service, opening up new arenas of innovetions, and altering the economics of distribution. It is for these reason that telecommunications becomes a strategic resource, capable of fundamentally altering firm's competitive positions and reshaping entire industries, as Keen, and, earlier, Cash and Konsynski [CK85] have noted..."

Synthétiquement, disons qu'il existe deux positions distinctes. L'une considère l'informatique comme un outil nécessaire à la domination des coûts et l'autre la considère comme une ressource stratégique. En résumé, la Banque se situe de nouveau entre deux des étapes identifiées par Nolan. Elle désire d'une part contrôler les coûts induits par les technologies de l'information et d'autre part les promouvoir et les intégrer à son organisation. Elle se trouve donc, sous cet éclairage, à mi-chemin entre les étapes de contrôle et d'intégration.

2.6. Les utilisateurs

L'étude historique du comportement des utilisateurs est plus délicate. Nous manquons sur ce point de données fiables. Par contre, pour ce qui est de la situation actuelle, nous disposons de faits précis et pertinents.

Une étude a été réalisée en 1989 par le Professeur Wilkin [WL89] de l'Université Libre de Bruxelles. Il tente, entre autres, de déterminer précisément le niveau de satisfaction du personnel de la Banque vis-à-vis des applications informatiques.

Nous proposons maintenant une rapide synthèse de cette étude :

Le Professeur Wilkin a développé un indice global de satisfaction. Cet indice est construit à partir de caractéristiques de l'application utilisée le plus fréquemment par le répondant.

"... L'échelle utilisée s'étend de 1 à 5. Un score élevé exprime un haut niveau de satisfaction; à l'inverse, un score faible indique un bas niveau de satisfaction. En d'autres termes, un score inférieur à trois exprime l'insatisfaction.

Cet indice global de satisfaction est, pour la Belgolaise, égal à 3,78. En regard des observations faites en Belgique, ce score est le plus élevé parmi ceux obtenus à ce jour. En revanche, il est inférieur aux normes établies pour les Etats-Unis..." [WL89].

Ensuite, il dégage les éléments qui contribuent au niveau relativement positif de satisfaction obtenu à la Belgolaise.

La figure II - 21 fournit, sous l'indice global, les valeurs des indices spécifiques pour la Belgolaise ainsi que la norme américaine correspondante.

"... Le niveau positif de satisfaction résulte pour l'essentiel de la facilité d'utilisation des diverses applications, de l'exactitude des données ainsi que du format sous lequel les "outputs" sont présentés.

En revanche, le contenu des informations apparaît comme le point le plus faible. En effet, bien qu'il indique une moyenne acceptable (+ 3,62), il équivaut à celui relevé dans une compagnie d'assurance au sein de laquelle cet aspect était jugé particulièrement insatisfaisant.

La qualité du caractère complet des informations dépend principalement de la mesure dans laquelle l'application tient compte des exigences et des attentes des utilisateurs.

Enfin, les utilisateurs ne sont pas satisfaits de la rapidité avec laquelle ils obtiennent les informations, c'est-à-dire du temps de réponse du système. Ceci est aussi le symptôme d'une charge excessive de l'ordinateur central.

Afin de permettre au lecteur d'avoir un aperçu global caractérisant la Belgolaise par rapport à d'autres organisations, la figure II - 23 présente, à titre d'information, les données de chaque indice pour chaque organisation, ainsi que la norme américaine pour chaque indice spécifique..." [WL89].

Enfin, le Professeur Wilkin mesure la satisfaction des utilisateurs en fonction des applications. Précisons que les indices de satisfaction les plus faibles sont enregistrés pour les applications les plus anciennes, à savoir : Crédocs (3,3), BZA (3,57), Change (3,67), Opérations Diverses (3,69) [Cfr Fig. II- 22].

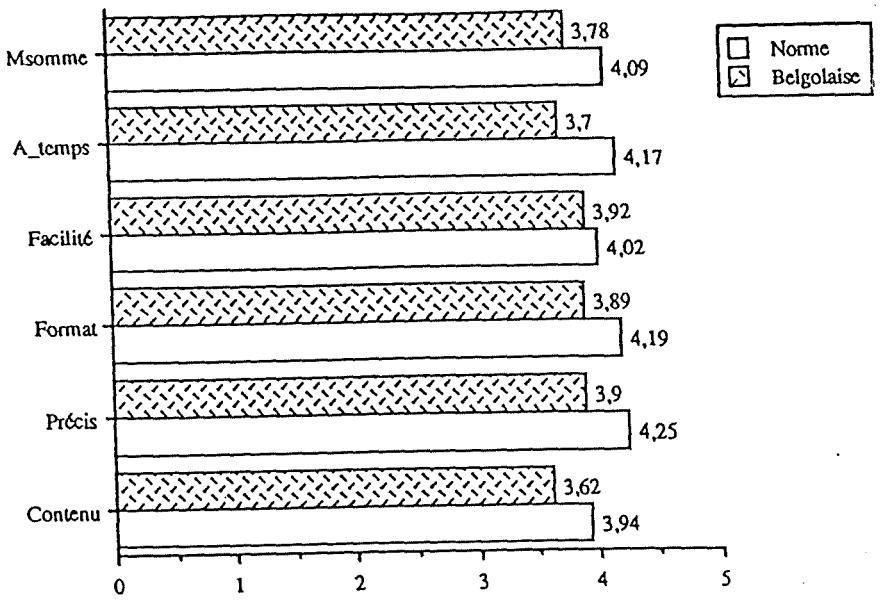


Fig. II.21.

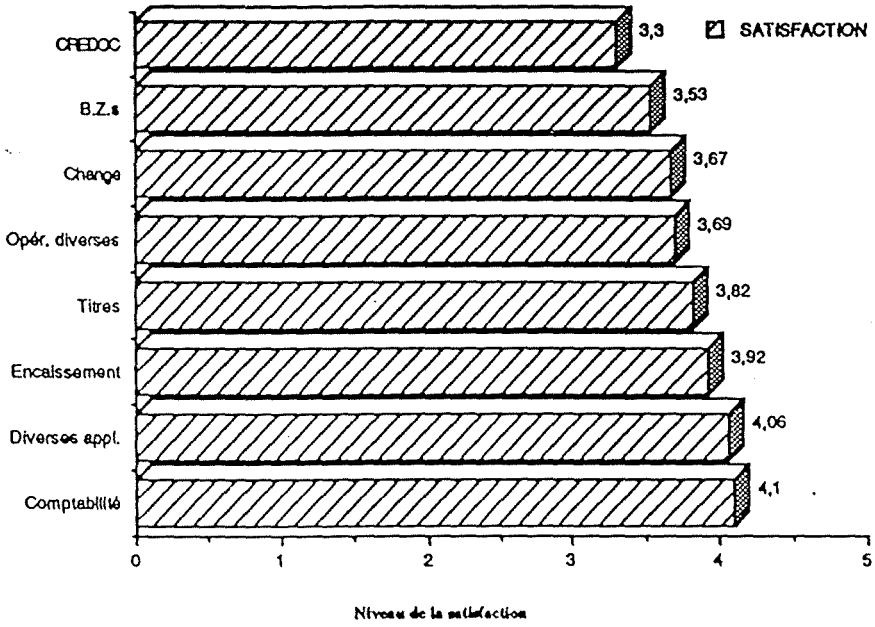


Fig. II.22.

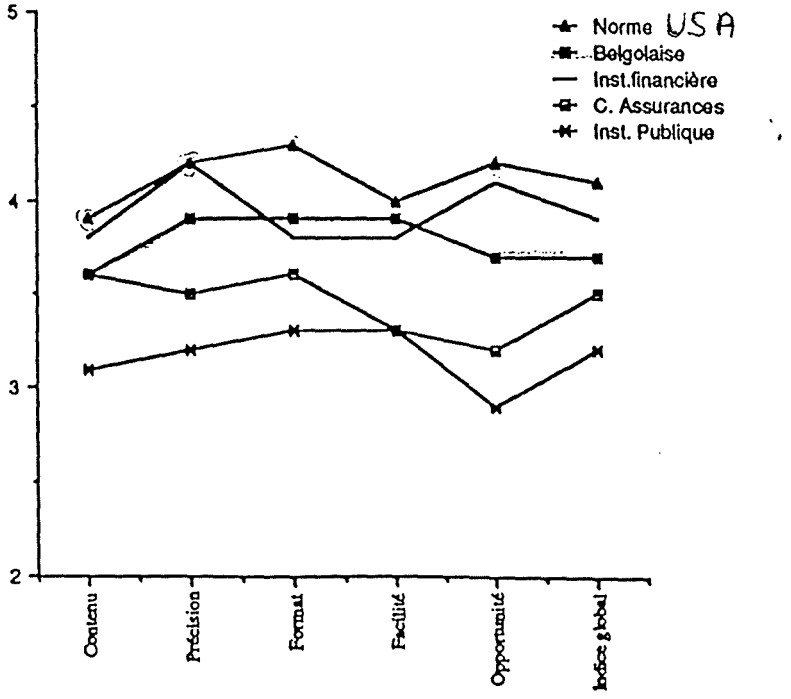


Fig. II.23.

Signalons d'autre part que l'ensemble du personnel de l'entreprise suit ou a suivi une formation spécifique à l'informatique. Malgré cela, on évalue [BL88] de 5 à 10 % la surconsommation des ressources CPU due à la mauvaise utilisation par assimilation insuffisante ou méconnaissance des facilités des systèmes. Ce dernier constat traduit une complexification constante des applications (intégration, modification, création de consultations de plus en plus élaborées). Seule, une formation permanente de l'ensemble du personnel peut remédier à cette situation.

De plus, dans chaque service, et parfois même dans chaque section, un agent est spécialement formé à la fonction de correspondant informatique.

Par ailleurs, la définition des spécifications et les tests d'acceptation des nouvelles applications passent obligatoirement par l'approbation d'une commission d'utilisateurs.

On peut affirmer en conclusion, que les utilisateurs sont globalement satisfaits des applications. Ils sont correctement formés, recyclés et supportés. La Belgolaise se trouve donc clairement en situation d'intégration quant à ses utilisateurs.

2.7. Les informaticiens

On sait que, dans son modèle, Nolan, décrit chronologiquement l'évolution du statut des informaticiens. Il est successivement mécanographe, programmeur et enfin, analyste.

Cette évolution est aussi perceptible dans cette entreprise. On constate en effet que la sélection du personnel informatique se porte de plus en plus sur des titulaires de diplômes universitaires. L'élévation du niveau de qualification et de formation suit, ici encore, celle prévue par Nolan.

2.8. Conclusion : entre le contrôle et l'intégration

En résumé et en conclusion de ce qui précède, la Banque se situe, pour presque chacune des variables, entre les étapes de contrôle et d'intégration. La Belgolaise est donc globalement à mi-chemin entre le contrôle et l'intégration.

Si d'une part, nos données s'avèrent exactes et que d'autre part, on accepte l'hypothèse évolutionniste de Nolan, nous pouvons essayer de prédire l'état global de l'organisation et celui des variables du modèle pour un futur proche.

Nous avons montré que la Banque était entre l'étape de contrôle et celle d'intégration. Il est donc probable qu'elle se dirige vers les étapes d'intégration et de gestion de la ressource information. On y constate généralement que les dépenses en informatique augmentent plus vite que le chiffre d'affaires. Du point de vue technologique, l'adoption de langage de haut niveau et l'utilisation intensive des réseaux et des PC sont aussi très fréquents. La direction mieux éclairée a une meilleure perception du phénomène informatique. Et enfin, les utilisateurs, mieux formés, l'exploitent de mieux en mieux.

3. CONCLUSION DU DEUXIEME CHAPITRE

Au terme de ce chapitre de description du système informatique de la Belgolaise, nous pouvons tirer les conclusions suivantes :

1) Le système informatique est calqué sur l'organisation qu'elle supporte, c'est-à-dire que :

   - il est de taille moyenne,
   - il utilise des technologies classiques,
   - les architectures logique et physique sont centralisées,
   - qualité : modularité, intégrité, unicité,
   - défaut : saturé (car au sommet de la gamme 43 IBM).

2) Les facteurs critiques de succès des applications informatiques sont :

   - la fiabilité,
   - l'optimisation du temps de réponse,
   - le fonctionnement optimal des (télé)communications.

3) La stratégie informatique est de développer des applications pour améliorer le service (vis-à-vis des clients et de la Banque). On constate donc :

   - un accroissement des coûts logiciels (développement, maintenance)
   - qui engendre une volonté de la Direction de contrôler ces coûts en augmentant la productivité des informaticiens.
   - obtenue par création et utilisation systématique de méthodes et d'outils d'aide au développement, à savoir ICE.

Nous venons de montrer pourquoi la Belgolaise désirait utiliser un environnement de développement. Dans le chapitre III, nous abordons l'étude approfondie de cet environnement "ICE".

CHAPITRE III : ETUDE APPROFONDIE DE L'ENVIRONNEMENT ICE

CHAPITRE III : Etude approfondie de l'environnement ICE

Rappelons que l'objet de ce mémoire est de présenter d'une part, la synthèse des aspects pertinents de l'informatique à la Belgolaise, et d'autre part, l'étude d'opportunité et les développements du projet ICE-PC que nous y avons réalisés.

Dans le chapitre II, nous avons présenté le système informatique de la Belgolaise. L'étude exhaustive de l'ensemble des aspects de l'informatique à la Banque dépasse, et de loin, le cadre de ce mémoire. Notre effort s'est donc entièrement porté sur l'examen approfondi de sa principale particularité : son environnement de développement "ICE".

De plus, la parfaite définition des concepts généraux de ICE est un préalable indispensable à toute adaptation de ce logiciel à un autre système informatique.

Le chapitre III propose donc une approche globale de l'environnement de développement ICE tel qu'il existe aujourd'hui à la Belgolaise. Le lecteur trouvera notamment une description précise des principes généraux qui le gouvernent, de même que lorsque cela a été possible, leurs justifications théoriques.

La description de cet environnement est réalisée par raffinements successifs. Des redondances existent mais nous espérons qu'elles devraient permettre au lecteur de mieux cerner l'essentiel de l'environnement étudié.

CHAPITRE III : Etude approfondie de l'environnement ICE

Ce troisième chapitre se compose des sections dont les buts sont les suivants :

1) Réaliser une première approche de l'environnement ICE qui permette au lecteur de se faire rapidement une idée de ce qu'est exactement ICE.

2) Exposer les raisons qui ont poussé la Belgolaise à développer son propre environnement de développement et les objectifs que poursuit ICE.

3) Présenter les principales fonctionnalités de l'environnement ICE.

4) Définir le langage de programmation des applications transactionnelles qui est l'une des particularités de l'environnement ICE.

5) Dégager les principes généraux qui forment les fondements et rendent efficace l'environnement ICE.

6) Déterminer les conséquences sur la production des logiciels qui résultent de l'utilisation de cet environnement particulier.

7) Présenter l'architecture de l'implémentation actuelle de l'environnement ICE sur le système informatique de la Belgolaise.

8) Une conclusion en forme d'évaluation globale de l'environnement ICE

1. PREMIERE APPROCHE

ICE est un ensemble de méthodes et d'outils qui sont mis à la disposition des analystes et programmeurs pour les aider dans leur travail. C'est donc un agrégat de méthodes et de programmes qui facilitent l'écriture des applications. C'est ce que l'on nomme dans le jargon informatique un environnement de développement de logiciels ou, plus simplement, un environnement de programmation ou encore un atelier logiciel. Ces divers synomymes seront indifféremment employés dans la suite.

Cet environnement est dédicacé à la conception d'une classe particulière d'applications. Les applications dont ICE facilite la réalisation sont à vocation transactionnelle [Cfr III - 4.1.]. C'est-à-dire qu'elles permettent d'effectuer une série de transactions du même type. Elles ne sont différentes que par les données qui leur sont fournies. On peut, par exemple, grâce à ICE, écrire un programme qui permette de saisir une suite de virements bancaires. Un autre programme pourrait mettre à jour le fichier des lignes de crédits.

ICE se compose de deux éléments principaux :

- Le premier d'entre eux est une méthode de conception des logiciels. Le principe de cette méthode est d'identifier un maximum de primitives de bases. Ces primitives sont les opérations que les développeurs utilisent très fréquemment ou qu'ils désirent très efficaces, ou encore très fiables.

  Par exemple une primitive ICE permet de faire apparaître à l'écran un message prédéfini. Le programmeur fournit un paramètre identifiant le message en question.

  A l'exécution, ICE se charge d'afficher ce message dans la langue de l'utilisateur. De plus, le comportement du programme sera fonction du degré de sévérité du message. Si c'est un "WARNING", le message est affiché puis le programme continue normalement, si c'est un "ERROR", le programme exige une correction avant de continuer, si c'est un "ABEND", le programme s'arrête automatiquement.

  D'autres primitives lisent ou écrivent sur un fichier, d'autres encore, impriment des listes, ...

Pour faire exécuter les primitives de bases, il suffit de les appeler par leur nom et de leur fournir les paramètres adéquats. Les primitives d'un même type sont regroupées au sein d'un même ensemble de primitives possédant la même syntaxe. ICE est, sous cet aspect des choses, un ensemble de boîtes à outils [22]. Il simplifie la vie des concepteurs puisqu'une fois qu'un outil existe, ils peuvent très simplement le réutiliser. Il suffit d'en connaître le mode d'emploi, la syntaxe.

- Le second élément de ICE est son langage de programmation. Celui-ci facilite grandement l'écriture de programmes transactionnels. Le programmeur décrit la transaction dans des paragraphes logiques prévus à cet effet [Cfr III - 4] L'enchaînement et la répétition de la transaction ne sont pas à charge du programmeur mais laissés aux bons soins de ICE.

  Pour supporter cette méthode et ce langage, il est nécessaire de disposer d'outils particuliers qui permettent d'exécuter sur une machine réelle les applications rédigées dans le langage ICE. Par exemple, dans sa configuration actuelle, l'environnement ICE dispose :

  - d'un précompilateur qui assure les tâches de traduction du langage ICE vers le langage COBOL,

  - d'une interface d'accès aux données, baptisée ICEFIC qui assure les fonctions de base d'un SGBD [16],

    par exemple : • la lecture
    • l'écriture
    • le verrouillage
    • ...

    ] d'enregistrements de fichiers

  - différents répertoires (fichiers, programmes, utilisateurs) qui rendent de nombreux services aux programmeurs,

  - ...

  De plus, différentes fonctions sont assurées de façon implicite par l'environnement ICE; parmi ces fonctions implicites se trouvent notamment :

  - la sécurité,
  - l'archivage,
  - l'interface homme-machine,
  - ...

Par exemple, la sécurité, si cruciale dans une institution bancaire, est entièrement prise en charge par les différents composants de ICE. Le développeur d'applications n'a jamais à s'en préoccuper. Elle ne nécessite aucune programmation particulière. La mise en place de la sécurité s'effectue en initialisant une série de paramètres dans les différents répertoires. Les problèmes d'identification et de confidentialité sont assurés de façon implicite. L'utilisateur du système n'aura accès qu'aux programmes par lesquels il a reçu une autorisation. Il en est de même pour les données et les transactions.

Ces différentes fonctions et primitives sont présentées plus en détail à la section 3 de ce chapitre.

Abordons maintenant aux objectifs que poursuit l'environnement ICE.

2. LES OBJECTIFS

Après ce premier aperçu de ICE, tentons de déterminer quels sont les objectifs pousuivis par cet environnement de développement.

Benington [BG56], et après lui Boehm [BM84], ont montré comment se répartissaient les coûts de production des logiciels. La distribution de ces coûts est la suivante :

- 50 % de tests unitaires (20 %) et d'intégration (30 %),
- 30 % de spécifications,
- 10 % de codage,
- 10 % de conception.

La moitié du temps est consacrée aux tests. On passe cinq fois plus de temps à vérifier si un programme est conforme aux spécifications qu'à le concevoir correctement dès le départ. Cette situation, loin d'être optimale, est encore plus grave si on la rapporte au coût total d'un projet informatique :

- 67 % de maintenance,
- 17 % de tests unitaires (7 %) et d'intégration (9,5 %),
- 10 % de spécifications,
- 3 % de codage,
- 3 % de conception.

La maintenance représente 67 % des coûts d'un projet par rapport à l'ensemble des coûts du cycle de vie. Maintenance et tests forment à eux seuls 84 % des frais de développement des applications. La discipline du génie logiciel fournit des méthodes et des outils qui permettent de diminuer la part de ces deux postes.

L'environnement ICE poursuit aussi ce but. Il tente de diminuer la quantité de ressource nécessaire au développement d'une application, c'est-à-dire de minimiser les coûts de production, d'exploitation et de maintenance. Ce sont ces préoccupations qui ont motivé l'élaboration de cet environnement de programmation.

Ce but est atteint par la réalisation de divers sous-objectifs :

Tout d'abord, débarrasser le programmeur d'un maximum de soucis d'ordre technique. L'informaticien peut se concentrer sur l'essentiel : la réalisation des spécifications fonctionnelles et, principalement, l'amélioration des services aux utilisateurs. C'est pourquoi, ICE se charge à sa place d'assurer de nombreuses contraintes techniques.

Par exemple, ICE dispense le programmeur de la gestion des écrans. En effet, sur le système IBM, pour afficher un écran, le programmeur doit utiliser les services d'un module appelé CICS [25]. La programmation en termes de ce module est très délicate. Sous ICE, il suffit au programmeur d'indiquer le numéro de l'écran à afficher ou à saisir.

Mais, ICE est aussi un langage de programmation fortement structuré et standardisé. Ses propriétés font qu'il facilite la lisibilité des applications. Les programmes sont dès lors plus faciles à corriger ou à modifier. De même, les programmeurs peuvent plus facilement passer d'une application à une autre parce que, dans chacune d'elles, ce sont les mêmes outils qui sont utilisés.

Enfin, cette méthode, par la réalisation de couches logicielles qui masquent l'existence des couches les plus basses, assure une plus grande portabilité des applications vis-à-vis des logiciels de base (OS [13] et SGBD [16]) ou du matériel.

Par exemple dans une autre société, l'adaptation de ICE sous VESAM à DLI [26] n'a nécessité que 2 semaines.

En conclusion, ICE simplifie la vie des analystes, des programmeurs et des utilisateurs en rendant les applications plus :

1) structurées,
2) lisibles,
3) portables,
4) standard.

En somme, il réalise les objectifs du génie logiciel, à savoir :

1) la fiabilité (validité et robustesse),
2) la "maintenabilité" :
   - modification,
   - extension,
   - contraction.
3) la réutilisabilité,
4) la portabilité,
5) l'efficacité,
6) la convivialité,
7) la sécurité.

3. LES FONCTIONNALITES

Nous venons de passer en revue les objectifs que poursuit l'environnement de développement "ICE". Nous allons maintenant détailler les principales classes de fonctions dont un programmeur ou un utilisateur dispose grâce à ICE, à savoir les fonctions :

1) d'accès aux données,
2) d'interfaces homme-machine,
3) de haut niveau,
4) implicites :
   - sécurité,
   - archivage,
   - logging.

3.1. L'accès aux informations

Nous avons vu [Cfr III - 1] que ICE assure l'ensemble des fonctions d'accès aux données. Nous en présentons maintenant quelques-unes parmi les plus importantes.

- Il facilite l'échange avec les bases de données. Pour des raisons de sécurité et de performance, ICE est aussi l'unique mode d'accès aux informations. Il possède une série de fonctions universelles d'accès fichiers. Citons par exemple la lecture, l'écriture, le blocage et le déblocage d'enregistrements, dont l'organisation peut être séquentielle ou indexée. Toutes ces opérations sont utilisables aussi bien par l'intermédiaire de clefs primaires que secondaires, multiples ou non.

- En outre, il autorise le verrouillage de plusieurs enregistrements simultanément, rendant par là-même impossible le trop tristement célèbre DEAD-LOCK [27]. De plus, il assure automatiquement le déverrouillage des enregistrements en cas "d'oublis" ou de fin anormale d'une transaction.

- Il dispense le programmeur de déclarer les fichiers qu'il utilise au sein de ses applications. Ces fichiers sont définis une fois pour toutes à l'aide d'un logiciel convivial, lui aussi fourni dans l'environnement ICE.

- Il fournit toujours, en réponse à une requête, un ensemble de codes d'erreur qui signale comment elle s'est déroulée.

  Par exemple, le code d'erreur "0" signale que la requête s'est bien déroulée.

  Toute autre valeur du code d'erreur signifie qu'il y a eu un problème dans la requête.

  - "4" signale que l'enregistrement est introuvable.
  - "8" signale la fin du fichier.
  - "12" signale que l'enregistrement est déjà verrouillé.
  - ...

- Enfin, il réalise de manière transparente pour le programmeur, les mécanismes vitaux des systèmes d'informations modernes tels les fonctions d'archivage, de roll-back et de recovery.

3.2. L'interface homme-machine

L'interface homme-machine qui, on le sait, tient une place de plus en plus importante dans l'ergonomie des logiciels est elle aussi grandement facilitée et standardisée par l'utilisation de ICE.

Durant l'exécution d'une application, les touches fonctions ont constamment la même signification. Par exemple, l'appui de la touche "PF3" déclenche toujours l'abandon du programme. Cela constitue d'ailleurs le seul et unique moyen pour l'utilisateur de signifier qu'il ne désire plus effectuer de transactions supplémentaires. "PF2" déclenche systématiquement l'abandon de la transaction courante. Enfin, "PF1" permet quant à elle de signaler la validité d'un écran. Les autres touches fonctions sont elles aussi porteuses d'une signification particulière. L'important est de bien comprendre que le programmeur n'a pas à assurer la gestion de ces touches. C'est ICE qui automatiquement réalisera toutes les instructions nécessaires.

Chaque écran d'une application contient toujours une zone destinée à y afficher des messages [Cfr III - 1]. Il suffit au programmeur d'indiquer l'identifiant du message prédéfini qu'il désire envoyer à l'écran, de même que la langue dans laquelle il le désire. ICE, comme d'habitude se charge du reste.

Enfin, la gestion des écrans, chose souvent délicate, est elle aussi prise en compte dans notre environnement. Un générateur d'écrans en facilite la création. Ces fonctions faciliteront non seulement leur réception, leur envoi et leur séquencement mais aussi la gestion dynamique du curseur et des attributs des différents champs des écrans. [Cfr Fig. III - 1 a. à Fig. III - h.]

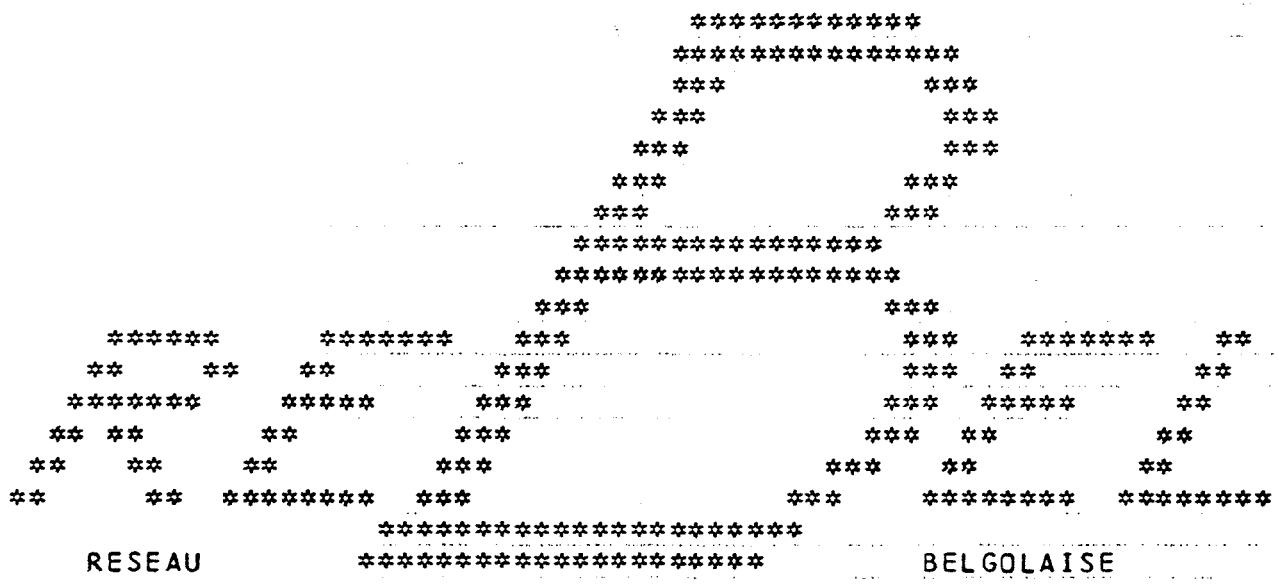
3.3. Les fonctions de haut niveau

Enfin, au-delà de ces primitives que l'on pourrait qualifier de bas niveau, ICE possède aussi un ensemble de fonctions, une boîte à outils de plus haut niveau. Ces outils sont dits de haut niveau car ils ont un sens non plus seulement pour l'ingénieur système, mais aussi pour le financier ou le comptable. Ces procédures peuvent réaliser des transactions complexes sur des ensembles de comptes et non plus seulement la mise à jour d'un élément isolé d'un fichier.

Par exemple, il existe une primitive ICE qui permet de débiter ou de créditer un compte d'une somme donnée à une "date-valeur" particulière. Cette notion de "date-valeur" est très importante. Elle signifie que l'opération est comptabilisée à une date X, mais que le compte est crédité ou débité à une date Y :

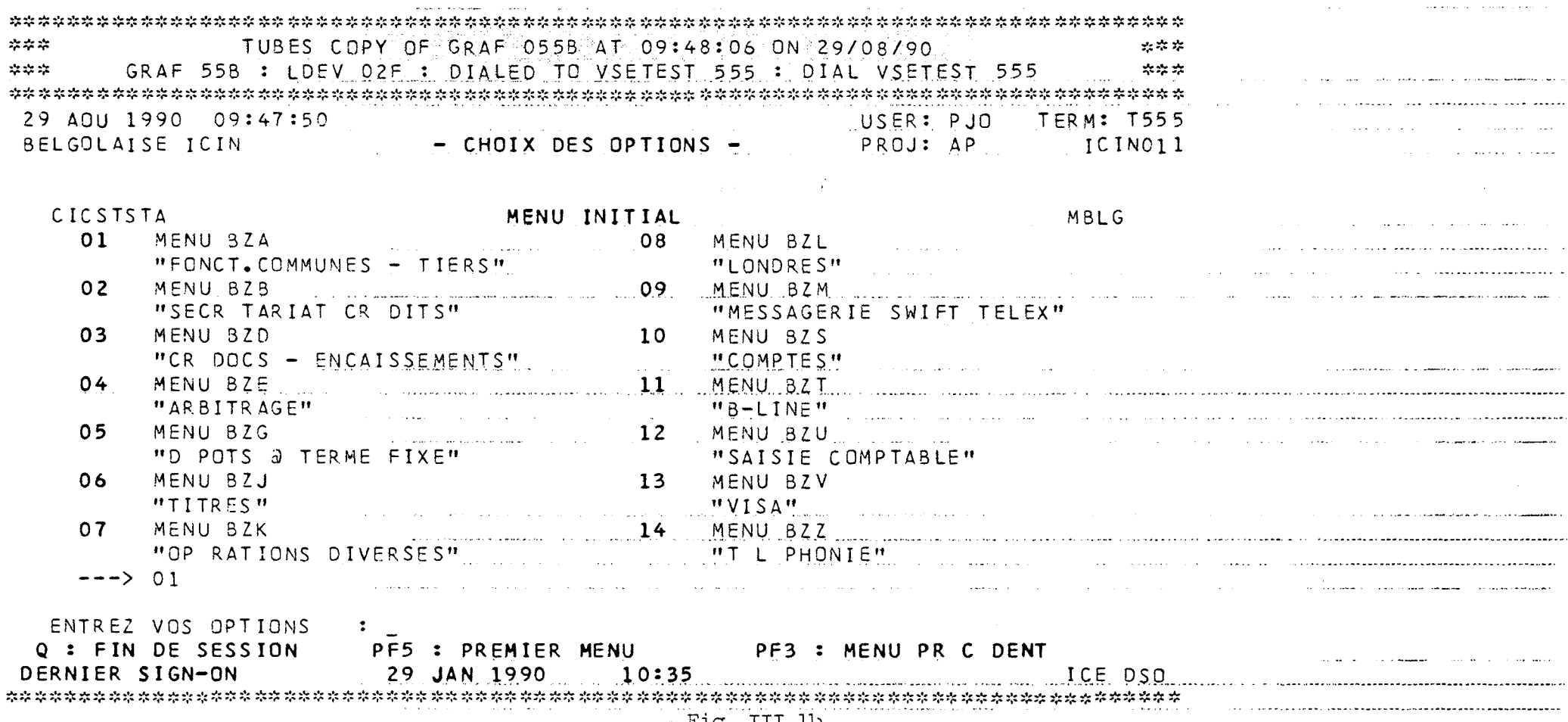
- Un client retire 100 Frs du compte 0001 le 07 janvier.
- Le compte 0001 est débité le 01 janvier de 100 Frs.
- Un client dépose 200 Frs du compte 0009 le 05 mars.
- Le compte 0009 est crédité le 15 mars de 200 Frs.

```
TUBES COPY OF GRAF 0558 AT 09:51:52 ON 29/08/90
GRAF 558 : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555
```




```
ICE 3.2.0          PROJECT : _                    USERID :
CICSTSTA           PASSWORD :                     NEW PASSWORD :          (PF1)
```

- Fig. III.1a. -


*** TUBES COPY OF GRAF 055B AT 09:48:06 ON 29/08/90 ***
*** GRAF 55B : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555 ***
29 AOU 1990 09:47:50 USER: PJO TERM: T555
BELGOLAISE ICIN - CHOIX DES OPTIONS - PROJ: AP ICIN011
CICSTSTA MENU INITIAL MBLG
01 MENU BZA
"FONCT.COMMUNES - TIERS"
02 MENU BZB
"SECR TARIAT CR DITS"
03 MENU BZD
"CR DOCS - ENCAISSEMENTS"
04 MENU BZE
"ARBITRAGE"
05 MENU BZG
"D POTS @ TERME FIXE"
06 MENU BZJ
"TITRES"
07 MENU BZK
"OP RATIONS DIVERSES"
08 MENU BZL
"LONDRES"
09 MENU BZM
"MESSAGERIE SWIFT TELEX"
10 MENU BZS
"COMPTES"
11 MENU BZT
"B-LINE"
12 MENU BZU
"SAISIE COMPTABLE"
13 MENU BZV
"VISA"
14 MENU BZZ
"T L PHONIE"
---> 01
ENTREZ VOS OPTIONS :
Q : FIN DE SESSION PF5 : PREMIER MENU PF3 : MENU PR C DENT
DERNIER SIGN-ON 29 JAN 1990 10:35 ICE DSO

- Fig. III.1b. -

```
**********************************************************************************
***              TUBES COPY OF GRAF 055B AT 09:48:27 ON 29/08/90               ***
***     GRAF 55B : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555         ***
**********************************************************************************
 29 AOU 1990  09:48:18                                  USER: PJO   TERM: T555
 BELGOLAISE ICIN            - CHOIX DES OPTIONS -       PROJ: SI       ICIN011

   CICSTSTA                      MENU SIGNALETIQUE                        MASI
     01   CR ATION D'UN TIERS                  08   ARCHIVAGE DES TIERS

     02   MODIFICATION D'UN TIERS

     03   AUTORISATION
          -------------------------
     04   VIREMENT DE CPTES @ CPTES
          -------------------------
     05   SUPPRESSION D'UN TIERS
          -------------------------
     06   INSTRUCTIONS SP CIALES
          APPLICATIONS
     07    DITION DES                      14   MENU DES CONSULTATIONS
          AVIS D'ACCOMPAGNEMENT                   SIGNAL TIQUES
     ---> 03

   ENTREZ VOS OPTIONS    : _
  Q : FIN DE SESSION     PF5 : PREMIER MENU        PF3 : MENU PR C DENT

**********************************************************************************
```

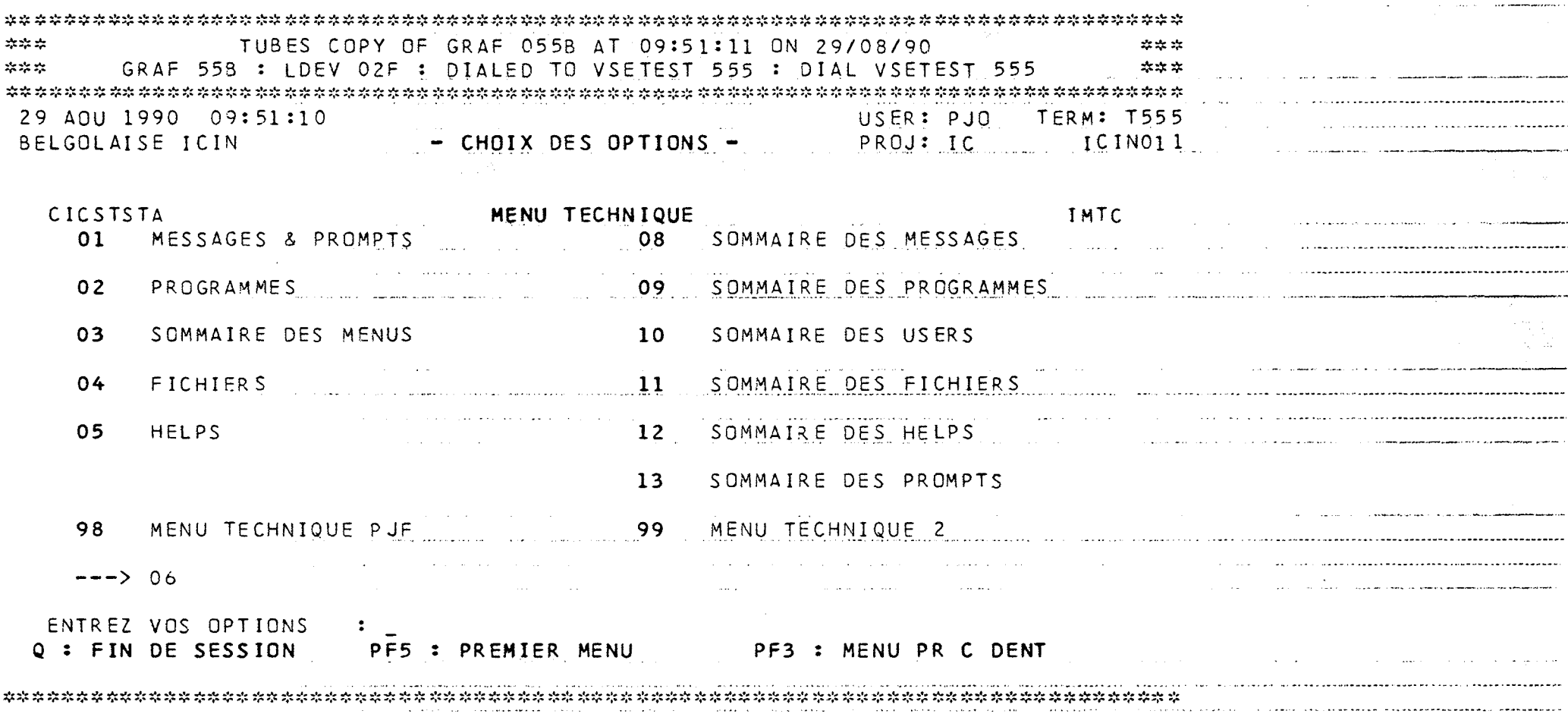
- Fig. III.1c. -

```
**********************************************************************************
***              TUBES COPY OF GRAF 055B AT 09:51:22 ON 29/08/90               ***
***     GRAF 55B : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555         ***
**********************************************************************************
 29 AOU 1990  09:51:16                                  USER: PJO   TERM: T555
 BELGOLAISE IC02        SOMMAIRE DES MESSAGES D'ERREUR   PROJ: IC       IC02021

 ACT MESSAGE S V LIBELL
 (X)
 _    AC090001 W FICHIER DES DEVISES INDISPONIBLE POUR LE MOMENT
      AC090002 W DEVISE INEXISTANTE DANS LE FICHIER DES DEVISES
      AC090003 W RATIOS INVALIDE
      AC120001 A NUMERO DE TIERS MANQUANT
      AC120002 A DEPASSEMEMT DE LA CAPACITE DE LA TABLE
      AC120003 A AUCUNE LIAISON POUR CE TIERS
      AC120004 A AUCUNE LIAISON NE VOUS EST ACCESSIBLE
      AC600001 W ENREGISTREMENT DU BZAD01 TABLE 20 VEROUILLE PAR UN AUTRE UTI
      AC600002 W TABLE 20 (DERNIER NUMERO DE TIERS ATTRIBUE) INEXISTANTE
      AC600003 W MISE A JOUR DE LA TABLE 20 DU BZAD01 IMPOSSIBLE
      AC610001 W ENREGISTREMENT DU FICHIER SIGNALETIQUE VERROUILLE
      AC610002 W NUMERO DE TIERS INEXISTANT
      AC610003 W NUMERO DE TIERS MIS A JOUR ET NON ENCORE AUTORISE
      AC610004 W MISE A JOUR DU FICHIER SIGNALETIQUE IMPOSSIBLE
      AXXX0000 E PAS D'ECRAN HELP POUR CETTE ZONE
      AXXX0001 E DESCRIPTION MANQUANTE

**********************************************************************************
```

- Fig. III.1d. -


*** TUBES COPY OF GRAF 055B AT 09:51:11 ON 29/08/90 ***
*** GRAF 558 : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555 ***
29 AOU 1990 09:51:10 USER: PJO TERM: T555
BELGOLAISE ICIN - CHOIX DES OPTIONS - PROJ: IC ICIN011
CICSTSTA MENU TECHNIQUE IMTC
01 MESSAGES & PROMPTS
02 PROGRAMMES
03 SOMMAIRE DES MENUS
04 FICHIERS
05 HELPS
08 SOMMAIRE DES MESSAGES
09 SOMMAIRE DES PROGRAMMES
10 SOMMAIRE DES USERS
11 SOMMAIRE DES FICHIERS
12 SOMMAIRE DES HELPS
13 SOMMAIRE DES PROMPTS
98 MENU TECHNIQUE PJF
99 MENU TECHNIQUE 2
---> 06
ENTREZ VOS OPTIONS :
Q : FIN DE SESSION PF5 : PREMIER MENU PF3 : MENU PR C DENT

- Fig. III.1e. -

```
**********************************************************************************
***              TUBES COPY OF GRAF 055B AT 09:50:42 ON 29/08/90               ***
***     GRAF 558 : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555         ***
**********************************************************************************
 29 AOU 1990  09:50:26                                  USER: PJO   TERM: T555
 BELGOLAISE A20S            RECHERCHE PAR MNEMONIQUE    PROJ: SI      A20S031
 CONSULT. FINANCIERE

                                                                  PAGE  001
   027834 92  PHILIPPE JOURION              JEAN

         -          603-5427834-49  BEF  R A
                    603-6427834-76  BEF  R A

**********************************************************************************
```

- Fig. III.1f. -

```
**********************************************************************************
***              TUBES COPY OF GRAF 055B AT 09:49:12 ON 29/08/90               ***
***     GRAF 55B : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555          ***
**********************************************************************************
 29 AOU 1990  09:49:03                                   USER: PJO   TERM: T555
 BELGOLAISE S309          SITUATION FINANCIERE D'UN CPTE PROJ: SI       S309011
 CONSULT. FINANCIERE



              CONSULTATION DU COMPTE NUM RO  : AAA



                                  PF5 CONSULTATION PAR MN MONIQUE ALPHA
 LE COMPTE N'EST PAS NUMERIQUE                                  ICE I54N
**********************************************************************************
```

- Fig. III.1g. -

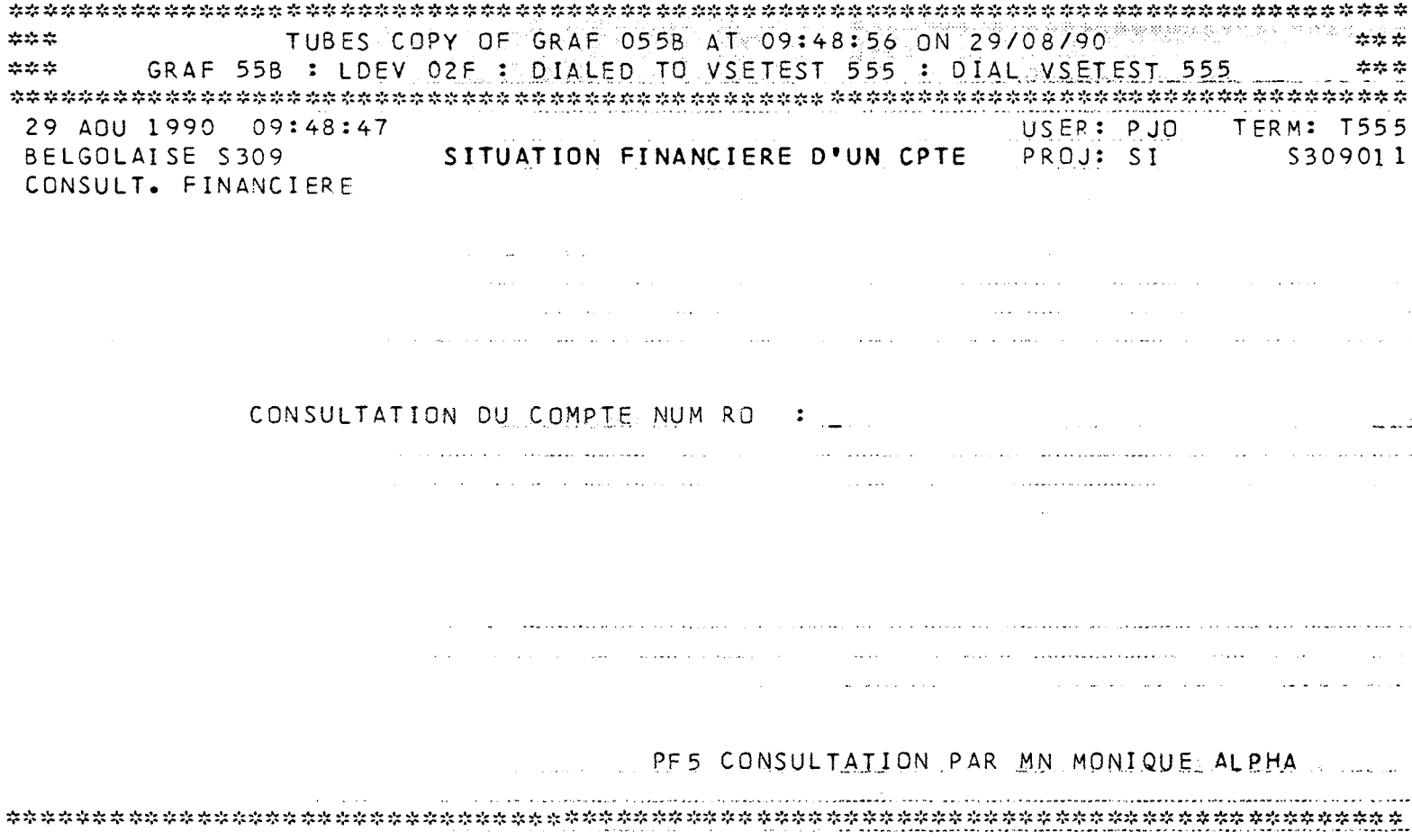

TUBES COPY OF GRAF 055B AT 09:48:56 ON 29/08/90
GRAF 55B : LDEV 02F : DIALED TO VSETEST 555 : DIAL VSETEST 555
29 AOU 1990 09:48:47
USER: PJO TERM: T555
BELGOLAISE S309
SITUATION FINANCIERE D'UN CPTE
PROJ: SI S309011
CONSULT. FINANCIERE
CONSULTATION DU COMPTE NUM RO :
PF5 CONSULTATION PAR MN MONIQUE ALPHA

Fig. III.1h.

De même, toute opération en dollar (devise) déclenche automatiquement la primitive de "gestion trésorerie dollar" (devise) qui met à jour l'échéancier dollar de la Banque.

3.4. Les fonctions implicites

Une série de fonctions, soit trop techniques, soit trop importantes pour être laissées à la discrétion du programmeur, sont assurées de façon implicite et donc systématique par l'environnement ICE.

Parmi ces fonctions, on trouve la sécurité et l'archivage.

a. La sécurité

La fonction de ICE est de soulager le programmeur d'un ensemble de considérations d'ordre technique. Les problèmes de sécurité appartiennent à cet ensemble et sont dès lors transparents à la programmation. Entièrement prise en charge par les différents composants de ICE, la sécurité est assurée de façon implicite.

Qu'il s'agisse d'accès aux programmes, aux transactions, ou encore de problèmes de confidentialité des données, tous ces aspects sont entièrement gérés par le noyau ICE qui vérifie à tout moment que l'utilisateur connecté est bien autorisé à effectuer les transactions qu'il commande.

Il est fondamental de comprendre que la mise en place de la sécurité ne nécessite aucune programmation particulière. Elle s'effectue par l'initialisation de paramètres dans les répertoires des fichiers des programmes et des utilisateurs [Cfr III - 7.3].

Cela induit une sécurité à trois niveaux :

Tout d'abord, sécurité "sign-on" au niveau de l'utilisateur. Tout accès à un programme ICE doit passer par une transaction qui, sur base du répertoire des utilisateurs, contrôle :

- son mot de passe,
- l'ancienneté de ce mot de passe,
- si le terminal qu'il utilise est autorisé,
- si l'heure correspond à une tranche horaire autorisée.

D'autre part, un utilisateur n'aura accès à un programme que s'il possède le code-privilège identique à celui requis pour ce programme. Evidemment, le code-privilège du programme est défini au niveau du répertoire des programmes.

Enfin, pour pouvoir accéder à un enregistrement donné d'un fichier, deux conditions doivent être remplies préalablement par l'utilisateur : posséder les codes de confidentialité et d'identification-client associés à cet enregistrement.

Le contrôle s'exerce, nous l'avons vu, à différents niveaux. ICE résoud donc les problèmes d'identification et de confidentialité. L'utilisateur du système n'aura accès qu'à certains programmes ou qu'à certaines données et ne pourra effectuer que certaines transactions.

b. L'archivage

L'archivage est un point critique en ce qui concerne l'intégrité des bases de données en cas de problème. Il existe une fonction implicite déclenchée par initialisation d'une variable dans le répertoire des fichiers qui active la mise à jour d'un fichier d'archivage.

A tout fichier correspond donc un fichier archivage qui contient toutes les versions successives des enregistrements qui ont modifiés et les identifiants des utilisateurs qui ont effectués ces modifications.

4. LE LANGAGE DE PROGRAMMATION

Les différentes fonctionnalités de l'environnement ICE que nous venons de passer en revue sont accessibles à partir d'un langage particulier : le langage de programmation transactionnelle ICE.

Nous proposons, avant de définir ce langage, de définir le concept de programme transactionnel.

4.1. Définition du concept de programme transactionnel

Avant de définir ce qu'est un programme transactionnel, il nous faut d'abord définir précisément ce qu'est une transaction.

La définition du concept de transaction que nous proposons ici nous est inspirée par Date [DATE].

"Une transaction est une unité de travail. Elle consiste en l'exécution d'une séquence d'opérations, commençant par, une opération particulière appelée "BEGIN TRANSACTION" et se terminant par, soit une opération appelée "COMMIT", soit une opération appelée "ROLL BACK". L'opération "COMMIT" signale la terminaison fructueuse de la transaction (l'unité de travail a été accomplie correctement). L'opération "ROLL BACK" sert à signaler une terminaison infructueuse de la transaction (l'unité de travail ne peut être accomplie correctement parce qu'un évènement exceptionnel est survenu; par exemple, une information indispensable ne peut être obtenue). Il est important de noter qu'il s'agit ici de la terminaison de la transaction et pas nécessairement du programme. L'exécution d'un programme peut correspondre à une séquence de plusieurs transactions, l'une après l'autre" [DATE].

On admettra dans la suite qu'un programme transactionnel est un programme dont le but est de réaliser une suite de transactions. Le schéma suivant montre l'exécution d'un programme transactionnel.

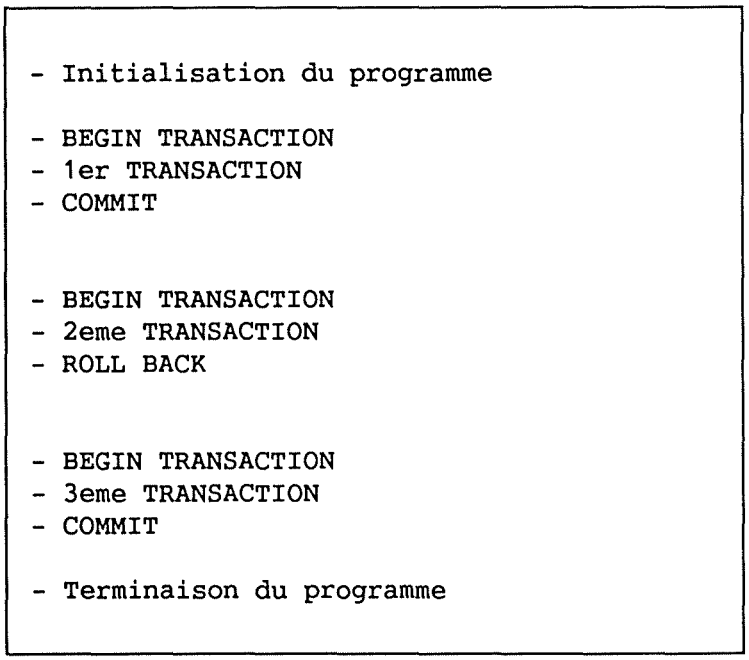


Fig. III - 2

"Les transactions ne peuvent pas être emboîtées. Un "BEGIN TRANSACTION" ne peut être exécuté que si aucune transaction n'est en cours. Inversement, un "COMMIT" ou un "ROLL BACK" ne peuvent être exécutés que si une transaction est en cours. Toutes les "opérations récupérables" doivent être exécutées entre les limites d'une transaction ("BEGIN TRANSACTION", "COMMIT" et "ROLL BACK").

Une opération est récupérable si elle peut être défaite ou refaite en cas d'échec (en d'autre mots, c'est une opération dont les entrées sont saisies en mode "LOG"). La mise à jour des bases de données et les messages d'I/O sont des opérations récupérables" [DATE].

Enfin, toute transaction peut aussi être décomposée en trois blocs ou paragraphes logiques décrivant respectivement :

1) l'initialisation de la transaction,
2) le corps de la transaction,
3) la terminaison de la transaction.

En résumé, tout programme transactionnel possède la structure suivante :

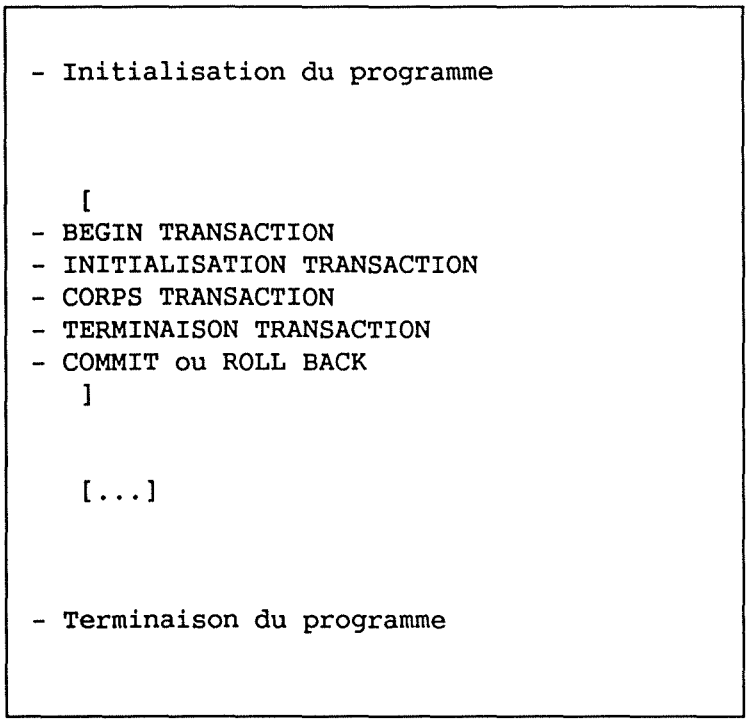


Fig. III - 3

4.2. La programmation transactionnelle en ICE

La description des programmes transactionnels en ICE est calquée sur la définition du concept de programme transactionnel que nous venons de proposer. Cette description est faite à l'aide d'une suite de paragraphes logiques particuliers.

Examinons d'abord quatre paragraphes : ICEIPR, ICEITR, ICETTR, ICETPR.

ICEIPR est le paragraphe descripteur de l'initialisation du programme. ICEITR est le paragraphe descripteur de l'initialisation de la transaction. ICETTR est le paragraphe descripteur de la clôture de la transaction. ICETPR est le paragraphe descripteur de la clôture du programme.

Par exemple, le programmeur d'application place les instructions d'initialisation du programme dans le paragraphe ICEIPR.

Ces paragraphes doivent apparaître dans cet ordre dans le texte du programme, juste après la "PROCEDURE DIVISION". Signalons aussi que le programmeur ne doit introduire ni "STOP RUN", ni "EXIT PROGRAM".

Entre les paragraphes ICEITR et ICETTR, c'est-à-dire entre l'initialisation et la clôture d'une transaction, vient s'intercaler une série de paragraphes descripteurs du corps de la transaction. Le nombre de ces paragraphes est toujours un multiple de trois parce qu'une transaction se compose d'une suite d'écrans. En effet, dans le langage ICE, chaque écran est décrit à l'aide de trois paragraphes :

Le premier décrit les instructions d'initialisation de cet écran. Il porte le nom "ICEXXX-I" (XXX représentant le numéro de l'écran).

Le second décrit les instructions de validation consécutives à la saisie de cet écran au terminal. Il porte le nom "ICEXXX-T".

Le troisième décrit les instructions à réaliser sur base des informations recueillies dans cet écran. Il porte le nom "ICEXXX-E".

En résumé, tout programme ICE est structuré de la façon suivante :

```
ICEIPR (Initialisation du programme)

   ICEITR (Initialisation de transaction)

   [
     ICEiii-I (initialisation écran iii)

     ICEiii-T (initialisation écran iii)

     ICEiii-E (initialisation écran iii)
   ]

   [...]

   ICETTR (Terminaison de transaction)

ICETPR (Terminaison du programme)
```

Fig. III - 4

Afin d'illustrer ce dernier concept, nous proposons le texte d'une application transactionnelle écrite dans le langage ICE [Fig. III - 4. a-e].

Cette application permet de consulter ou de modifier le nom du titulaire d'un poste téléphonique sur base du numéro de ce poste.

5. LES PRINCIPES GENERAUX

Maintenant que nous connaissons les objectifs et les fonctionnalités et le langage de cet environnement, attachons-nous dans cette section à découvrir et à préciser quels sont les principes généraux qui forment les fondements de ICE. Chaque fois que nous aurons dégagé un de ses fondements, nous le confronterons aux recommandations théoriques existant à ce sujet dans la littérature.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. Z20M.
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                                                               *
      *   TITULAIRES TELEPHONIE : DETAIL                              *
      *   ------------------------------                              *
      *       - MAP 1 : DETAIL                                        *
      *                                                               *
      *   PGM APPELE PAR Z20S EN TASK-SWITCH                          *
      *                                                               *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
       AUTHOR. PJO.
      *
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      *
      *    ZONES A SAUVER
      *    --------------
       01  ICEREC.
           03  WORKSV.
               05  LISCOM      PIC X.
               05  LISCAD      PIC X.
               05  LISSER      PIC X.
               05  EXTENS.
                   07  EXTEN1      PIC X VALUE SPACE.
                   07  EXTEN3      PIC 9(3).
      *
      *    DEFINITION DES ECRANS
      *    - - - - - - - - - - -
           COPY Z20M01  REPLACING Z20M01I BY ICE001.

      *
      *    DEFINITION DES FICHIERS
      *    - - - - - - - - - - - -
           COPY BZZD20.

      *
      *    INFORMATIONS ENVOYEES ENTRE MODULES
      *    - - - - - - - - - - - - - - - - - -
      *        MODULE Z20M          --> MODULE Z20M
      *    ACTION + CLE

           03  SAVZON.
               05 SAVACT   PIC  X.
                  88  CONSULT VALUE 'C'.
                  88  MODIF   VALUE 'M'.
                  88  AJOUT   VALUE 'A'.
                  88  SUPPR   VALUE 'S'.
               05 SAVNUM   PIC X.
                  88  CLE02   VALUE '2'.
                  88  CLE03   VALUE '3'.
                  88  CLE04   VALUE '4'.
               05 SAVCLE       PIC X(10).
               05 SAVCLE22     REDEFINES SAVCLE.
```

- Fig. III.4a. -

```
                07 SAVCLE2      PIC X(4).
                07 FILLER       PIC X(6).
             05 SAVCLE33    REDEFINES SAVCLE.
                07 SAVCLE3      PIC X(5).
                07 FILLER       PIC X(5).
             05 SAVCLE44    REDEFINES SAVCLE.
                07 SAVCLE4      PIC X(10).
             05 EXTENS.
                07 EXTEN1       PIC X.
                07 EXTEN3       PIC X(3).
             05 MATRIC       PIC X(5).

      01  WORKER.
          03  RECPER       PIC 9(3).
          COPY BZZD01.

       PROCEDURE DIVISION.

       ICEIPR.
          MOVE ICEZON TO SAVZON.
          IF CONSULT
             MOVE 'CONSULTATION DES TITULAIRES' TO ICELPG RLC,30,'C'
          ELSE
             MOVE 'MISE-A-JOUR DES TITULAIRES'  TO ICELPG RLC,30,'C' .
          GO TO ICE.

      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                  DEBUT DE TRANSACTION                           *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
       ICEITR.
          MOVE EXTENS OF SAVZON TO EXTENS OF CLEFIC OF BZZD20.
          MOVE MATRIC OF SAVZON TO MATRIC OF CLEFIC OF BZZD20.
          IF MODIF
             CALL 'ICEFIC' USING ZREADIND BZZD20 ZLOCK
          ELSE
             CALL 'ICEFIC' USING ZREADIND BZZD20.
          IF RETCOD OF BZZD20 = 12
             MOVE 'XXXX0043' TO ICEMES
      *      ENREGISTREMENT BLOQUE
             PERFORM ICE.
          IF RETCOD OF BZZD20 = 0 AND AJOUT
             MOVE 'XXXX0021' TO ICEMES
      *      ENREGISTREMENT EXISTANT
             PERFORM ICE.
          IF RETCOD OF BZZD20 NOT = 0 AND NOT AJOUT
             MOVE 'XXXX0045' TO ICEMES
      *      ENREGISTREMENT INEXISTANT
             PERFORM ICE.

          GO TO ICE.

      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                    M  A  P  1                                   *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
       ICE001-I.
```

- Fig. III.4b. -

```
           MOVE SPACES              TO ICE001.
           MOVE ICECRS TO NOMALPI OF ICE001.
           IF AJOUT
              MOVE 'AJOUT' TO ICEHD3
              MOVE SPACE TO ENREGI OF BZZD20.
           IF CONSULT
              MOVE 'CONSULTATION' TO ICEHD3
              PERFORM PRO-SCR1.
           IF MODIF
              MOVE 'MISE-A-JOUR' TO ICEHD3.
           PERFORM FILL-SCR1.
           GO TO ICE.

       ICE001-T.
           IF CONSULT
              GO TO ICE.
           MOVE 'Y' TO ICEREF.

           IF SUPPREI OF ICE001 NOT = '0' AND NOT = 'N'
              MOVE ICEREQ TO SUPPREI OF ICE001
              MOVE 'ZXXX0001' TO ICEMES
      *       DOIT ETRE O OU N
              PERFORM ICE.
           IF SUPPREI OF ICE001 = '0'
              MOVE 'S' TO SAVACT
              MOVE 'SUPPRESSION' TO ICEHD3
              MOVE EXTEN3 OF SAVZON TO EXTENSI OF ICE001
              GO TO ICE.

           MOVE EXTENSI OF ICE001  NUM,3   TO EXTEN3 OF WORKSV.
           MOVE EXTEN3  OF WORKSV          TO EXTENSI OF ICE001.

           MOVE '01' TO NUMTAB OF BZZD01.
           MOVE SERVICI OF ICE001          TO CLETAB OF BZZD01.
           CALL 'ICEFIC' USING ZREADIND BZZD01.
           IF RETCOD OF BZZD01  = 0
              MOVE ABGSER OF BZZD01 TO ABGSERI OF ICE001
           ELSE
              MOVE SPACES TO ABGSERI OF ICE001
              MOVE ICEREQ TO SERVICI OF ICE001
              MOVE 'ZXXX0002' TO ICEMES
      *       SERVICE INEXISTANT
              PERFORM ICE.

           IF RECPERI OF ICE001 NOT = SPACES
              MOVE RECPERI OF ICE001  NUM,3  TO RECPER OF WORKER
              MOVE RECPER OF WORKER TO RECPERI OF ICE001.

           MOVE LISCOMI OF ICE001  SWI  TO LISCOM OF WORKSV.
           MOVE LISCADI OF ICE001  SWI  TO LISCAD OF WORKSV.
           MOVE LISSERI OF ICE001  SWI  TO LISSER OF WORKSV.
           MOVE LISCOM OF WORKSV TO LISCOMI OF ICE001  SWI .
           MOVE LISCAD OF WORKSV TO LISCADI OF ICE001  SWI .
           MOVE LISSER OF WORKSV TO LISSERI OF ICE001  SWI .
           GO TO ICE.
```

- Fig. III.4c. -

```
       ICE001-E.
            MOVE 'Y' TO ICEETR.
            GO TO ICE.

      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                    FIN DE TRANSACTION                           *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
       ICETTR.
            IF AJOUT
               PERFORM FILL-FILE
               CALL 'ICEFIC' USING ZWRITE BZZD20.
            IF SUPPR
               CALL 'ICEFIC' USING ZDELETE BZZD20.
            IF MODIF
               CALL 'ICEFIC' USING ZDELETE BZZD20
               PERFORM FILL-FILE
               CALL 'ICEFIC' USING ZWRITE BZZD20
               IF RETCOD OF BZZD20 NOT = 0
                  MOVE 'ZXXX0099' TO ICEMES
      *           ENREGISTREMENT DOUBLE  (ABEND)
                  MOVE 'BZZD20' TO ICEL24
                  PERFORM ICE.
            MOVE 'Y'                    TO ICEEPR.
            GO TO ICE.

       ICETPR.
            GO TO ICE.

      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                    P E R F O R M S                              *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*

       PRO-SCR1.
            MOVE ICEASK TO EXTENSI OF ICE001.
            MOVE ICEYEL TO EXTENSI OF ICE001.
            MOVE ICEASK TO NOMALPI OF ICE001.
            MOVE ICEYEL TO NOMALPI OF ICE001.
            MOVE ICEASK TO NOMTITI OF ICE001.
            MOVE ICEYEL TO NOMTITI OF ICE001.
            MOVE ICEASK TO SERVICI OF ICE001.
            MOVE ICEYEL TO SERVICI OF ICE001.
            MOVE ICEASK TO FONCTII OF ICE001.
            MOVE ICEYEL TO FONCTII OF ICE001.
            MOVE ICEASK TO RECPERI OF ICE001.
            MOVE ICEYEL TO RECPERI OF ICE001.
            MOVE ICEASK TO LISCOMI OF ICE001.
            MOVE ICEYEL TO LISCOMI OF ICE001.
            MOVE ICEASK TO LISCADI OF ICE001.
            MOVE ICEYEL TO LISCADI OF ICE001.
            MOVE ICEASK TO LISSERI OF ICE001.
            MOVE ICEYEL TO LISSERI OF ICE001.
            MOVE ICEASK TO SUPPREI OF ICE001.
            MOVE ICEYEL TO SUPPREI OF ICE001.
```

Fig. III.43. -

```
       FILL-SCR1.                                                       Z
           MOVE EXTEN3 OF SAVZON              TO EXTENSI OF ICE001.     Z
           MOVE MATRIC OF SAVZON              TO MATRICI OF ICE001.     Z
           MOVE NOMALP OF CLEF04 OF BZZD20 TO NOMALPI OF ICE001.        Z
           MOVE SERVIC OF CLEF03 OF BZZD20 TO SERVICI OF ICE001.        Z
           MOVE NOMTIT            OF BZZD20 TO NOMTITI OF ICE001.       Z
           MOVE '01' TO NUMTAB OF BZZD01.                               Z
           MOVE SERVIC OF CLEF03 OF BZZD20 TO CLETAB OF BZZD01.         Z
           CALL 'ICEFIC' USING ZREADIND BZZD01.                         Z
           IF RETCOD OF BZZD01  = 0                                     Z
              MOVE ABGSER OF BZZD01 TO ABGSERI OF ICE001                Z
           ELSE                                                         Z
              MOVE SPACES              TO ABGSERI OF ICE001.            Z
           MOVE FONCTI            OF BZZD20 TO FONCTII OF ICE001.       Z
           MOVE RECPER            OF BZZD20 TO RECPERI OF ICE001.       Z
           MOVE LISCOM OF BZZD20 TO LISCOMI OF ICE001  SWI .            Z
           MOVE LISCAD OF BZZD20 TO LISCADI OF ICE001  SWI .            Z
           MOVE LISSER OF BZZD20 TO LISSERI OF ICE001  SWI .            Z
           MOVE 'N'               TO SUPPREI OF ICE001.                 Z
                                                                        Z
       FILL-FILE.                                                       Z
           MOVE EXTENS  OF WORKSV TO EXTENS OF CLEFIC OF BZZD20         Z
                                     EXTENS OF CLEF02 OF BZZD20         Z
                                     EXTENS OF CLEF03 OF BZZD20.        Z
           MOVE NOMALPI OF ICE001 TO NOMALP OF CLEF02 OF BZZD20         Z
                                     NOMALP OF CLEF04 OF BZZD20.        Z
           MOVE SERVICI OF ICE001 TO SERVIC OF CLEF03 OF BZZD20.        Z
           MOVE MATRICI OF ICE001 TO MATRIC OF CLEFIC OF BZZD20.        Z
           MOVE FONCTII OF ICE001 TO FONCTI OF BZZD20.                  Z
           MOVE RECPERI OF ICE001 TO RECPER OF BZZD20.                  Z
           MOVE LISCOM  OF WORKSV TO LISCOM OF BZZD20.                  Z
           MOVE LISCAD  OF WORKSV TO LISCAD OF BZZD20.                  Z
           MOVE LISSER  OF WORKSV TO LISSER OF BZZD20.                  Z
           MOVE NOMTITI OF ICE001 TO NOMTIT OF BZZD20.                  Z
```

- Fig. III.4e. -

5.1. La boîte à outils

Nous savons que ICE se base, entre autres, sur le principe de la boîte à outils. Les outils, une fois créés, sont mis à la disposition de tous les programmeurs.

Que dit la théorie au sujet de ce concept ? Selon A. van Lamweerde, le principe de la boîte à outils est le suivant :

"Il s'agit d'une collection d'outils, souples, compatibles et complémentaires, dans laquelle l'analyste/programmeur peut aller puiser les outils paraissant les plus adéquats (...). Ces outils sont donc passifs : ils n'induisent pas une méthodologie particulière, et l'initiative est laissée à leur utilisateur qui a donc une grande liberté de mouvements [VLWD]".

"La caisse à outils s'intègre de façon naturelle dans un atelier logiciel ou environnement de production de logiciels (...) L'illustration la plus notoire de ce concept est le Programmer Workbench des Bell Labs, décrit dans Dolotta et Mashey [DM76] et Ivie [IV77]; les principes de base qui y sont suivis consistent à séparer le développement de logiciels de leur exploitation, et à fournir des interfaces uniformisées aux différents responsables du développement (...). La cause principale du large succès connu par cet environnement est son caractère adaptable et non spécialisé (...). (UNIX [6]) offre à l'utilisateur un ensemble très grand de programmes indépendants et à forte cohésion interne (c'est-à-dire réalisant une seule fonction). Une combinaison bien choisie de ces programmes permet parfois à l'utilisateur de faire l'économie d'écrire un programme pour son problème [VLWD]".

D'autre part, Sommerville tient le même raisonnement à propos des boîtes à outils :

"Subroutine libraries are probably the earliest instance of organized program sharing. If a general useful subroutine is prepared by one individual, that routine can be entered in a public library of subroutines and any other user may refer to that subroutine in his or her program..." [SM82].

La puissance d'une boîte à outils provient de ce qu'un programmeur "utilise" les outils mis à sa disposition.

La relation "utilise" est le fondement du concept de boîte à outils. Une boîte à outils n'est en effet qu'un ensemble cohérent de programmes susceptibles d'être utilisés par des programmes de niveaux supérieurs.

Par définition, si "A" utilise "B" alors :

1) "A" est nettement plus simple du fait qu'il utilise "B".
2) "B" n'est pas beaucoup plus compliqué du fait de ne pas utiliser "A".
3) Il existe un sous-système utile contenant "B" et pas "A".
4) Il n'existe pas de sous-système utile contenant "A" et pas "B".

Signalons que la structuration d'un système à l'aide de la relation "utilise" :

1) permet l'élimination des redondances fonctionnelles,

2) permet la réalisation de niveau d'abstraction qui implique la factorisation du travail,

3) facilite le développement incrémental par emboîtement de sous-systèmes utiles,

Ces propriétés s'avèrent particulièrement intéressantes durant trois phases du cycle de vie d'un logiciel :

1) Durant la phase de conception, le responsable d'un module ne doit rien connaître des modules appelants. Les modules appelants ne doivent connaître que les spécifications externes du module appelé. Et enfin, la relation "utilise" favorise la réutilisabilité des modules appelés car ces derniers ne font aucune hypothèse sur ceux qui les appellent.

2) Durant la phase de validation, l'existence de la relation "utilise" permet la factorisation des tests. Un composant appelé par différents modules ne doit être testé qu'une seule fois.

3) Durant la phase de maintenance, la réalisation de niveaux d'abstraction induite par la relation "utilise", implique la modification des seuls niveaux concernés par les changements.

5.2. L'interface cachant des secrets

Nous venons de voir comment l'environnement ICE favorisait la factorisation du travail par la création de boîtes à outils. Mais sous ICE, les outils ne sont pas rassemblés n'importe comment : les outils d'un même type sont réunis au sein d'une même boîte.

Il existe donc plusieurs boîtes à outils, chacune assurant des fonctions d'un type particulier.

Par exemple, tous les outils d'accès aux données sont réunis dans une boîte à outils particulière appelée "ICEFIC" [30].

Mais "ICEFIC" est plus qu'une boîte à outils, c'est aussi l'interprète par lequel passe le flux des données. Il assure l'échange entre 2 programmes : les applications et la base de données.

ICEFIC est donc une interface entre 2 composants logiciels. Voyons maintenant plus précisément la définition et l'intérêt de ce mécanisme d'interface.

"It is often assumed that the interface between two sofware components may be described by describing the format of the information that they exchange. This is a gross oversimplification which has resulted in a great many expensive errors. A complete description of the interface must include a statement of all of the assumptions that each component makes about the other. Anything less is not a complete description of the ways (intended and uninteded) that the two components might interact..." [PR77].

L'intérêt de ce mécanisme réside dans le fait que la façon dont une interface réalise une fonction ne doit pas être connue par les programmeurs de l'application. Seule la syntaxe de la requête a de l'importance. La représentation physique des données est complètement transparente pour le programmeur.

Que les informations soient sur bande ou sur disque, dans un fichier séquentiel ou indexé, elles n'apparaissent pas dans le texte des applications. ICE cache donc de l'information. On dit qu'il possède une forte capacité à cacher des secrets. Cette capacité à cacher des secrets a été érigée en principe général dans l'environnement ICE.

En anglais, ce principe se traduit par "information hiding". Examinons les recommandations que fait Parnas à ce sujet, dans son texte "Use of Abstract Interfaces in the Development of Software for Embedded Computer System" [PR77]. Il définit le principe de "information hiding" à l'aide de trois caractéristiques :

1) "Identify a list of design decisions for which change cannot be ruled out (data structure, algorithms, ...)".

2) "Make each design decision the 'secret' of one module...".

3) "Desing the module interface. The interface consists of the 'subprograms' needed by the module user in order to make use of the module's data structure and algorithms without knowing the design decision that is being hidden. This interface is so designed that it can be kept unchanged even if the data structure or algorithm must be revised..." [PR77].

Le principe de dissimulation de l'information est aussi désigné par Sommerville [SM82] comme fondamental dans la conception de logiciel.

Les principes d'interfaçage et de dissimulation de l'information donnent naissance à des systèmes dont la structure est représentée ici :

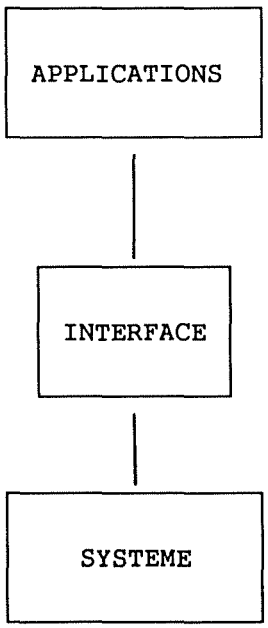


Fig. III - 5

Le rectangle supérieur représente les applications. Elles sont écrites en terme de services de l'interface. Elles sont donc indépendantes des caractéristiques changeantes du système. La façon dont l'interface assure la conversion des informations entre le système et les applications constitue le secret de l'interface.

5.3. Le langage de haut niveau

ICE est un environnement de développement d'application. Ces applications sont écrites dans un langage particulier.

Ce langage particulier est dérivé du langage COBOL dont il tente de corriger les défauts. Jackson les énumère dans [SE76] :

1) la dépendance excessive par rapport à la machine,
2) les anomalies syntaxiques à tous les niveaux,
3) la faible capacité pour définir de nouvelles primitives,
4) l'absence d'une philosophie cohérente guidant la conception des programmes.

En ce qui concerne le premier de ces défauts, l'ensemble des dépendances vis-à-vis de la machine sont rassemblées en COBOL dans la partie 'ENVIRONMENT DIVISION' des déclarations. Cependant, ces dépendances restent encore importantes. ICE, en supprimant l'écriture de cette division, diminue de la sorte de façon très importante ces dépendances.

En COBOL standard, les seuls mécanismes permettant de définir de nouvelles primitives sont ceux du "COPY" et du "PERFORM". Ces deux mécanismes ne permettent cependant pas la compilation séparée des primitives.

Selon Sommerville : "Without the facility of independent compilation, a language cannot be considered as a viable language for software engineering [SM82]."

C'est pourquoi, depuis longtemps, la plupart des compilateurs supportent l'instruction "CALL". Elle permet de créer les librairies de procédures et de sous-programmes qui peuvent être implémentées séparément. C'est souvent comme cela que l'on a adjoint à COBOL de puissants SGBD.

ICE est lui aussi bâti sur le même principe de l'extension à outrance du COBOL par l'addition de primitives. Ces dernières sont implémentées séparément. Pour des raisons de performance, les primitives ICE sont écrites en ASSEMBLEUR. Elles sont ensuite compilées séparément. Le programmeur les utilise à l'aide du mécanisme du "CALL".

En plus de ces facilités d'extension du COBOL, le langage ICE bénéficie d'une autre propriété. Il facilite énormément la programmation transactionnelle par l'utilisation de paragraphes particuliers [Cfr III - 4].

Grâce aux différents mécanismes, d'addition de primitives et de recombinaison de paragraphes, le langage ICE devient petit à petit un langage de plus haut niveau que COBOL.

Enfin, Sommerville [SM82] signale aussi qu'une bonne présentation d'un programme en améliore la lisibilité. Tout programme ICE, possède la même présentation. Les transactions sont toujours décrites de la même façon. D'autre part, un programmeur peut définir, dans son éditeur de texte, toute une série d'attributs particuliers pour ses écrans (couleurs, sur-brillance...). ICE améliore donc la présentation et la lisibilité des programmes.

5.4. La standardisation systématique

Un autre aspect important de ICE est qu'il assure toute une série de services de façon implicite. La sécurité et l'archivage sont deux bons exemples. ICE s'en charge sans que le programmeur ait même à s'en soucier. Il n'est plus question ici d'outils auxquels le programmeur fait appel. Ces fonctions sont assurées une fois pour toutes de façon transparente par le système. Cette "glasnost" informatique a pour conséquence la standardisation des applications. Mais si ICE induit, par son utilisation, une standardisation systématique (sécurité, interface homme-machine), cette dernière en est aussi l'un des principes constructeurs fondamentaux.

Par exemple, on trouve dans le texte d'un programme certains types d'objets particuliers, à savoir : les constantes, les variables, les procédures, les fonctions, etc... Dans son ouvrage [SM82], Sommerville estime que le nom de ces différents objets doit être en relation avec les objets du monde réel qu'il représente. Il fournit des règles d'abréviation permettant de dériver, à partir du nom des objets dans la réalité, le nom des objets qui les représentent dans le programme.

L'environnement ICE souscrit à ces recommandations. Tout nom d'un objet doit avoir six caractères. Il existe plusieurs règles d'abréviation selon le type d'objet. Par exemple pour un fichier, les deux premières lettres représentent le nom de l'organisation dans laquelle fonctionne le système. Dans le cas de la Belgolaise, il s'agit des lettres 'BZ'. Les deux suivantes correspondent aux initiales de l'application qui utilise ce fichier. Et enfin, les deux derniers caractères identifient le fichier. Cette façon toujours identique de choisir un nom est un autre exemple qui montre que la standardisation est un principe directeur de ICE.

Par exemple :

B.Z.	C = Change V = Visa O = Opérations Divers	D = indexé T = séquentiel W = travail	01 01 99

BZCD01 fichier indexé n° 01 du Change
BZVW02 fichier de travail n° 02 du Visa
BZOT01 fichier séquentiel n° 01 des Opérations Diverses
...

5.5. Les fonctions de haut niveau

Nous savons déja que ICE rend différents types de services. Nous connaissons les services d'accès aux données, ceux qui concernent la sécurité mais il en existe aussi qui permettent le support de fonctionnalités bancaires de plus haut niveau. Par exemple, la mise à jour des échéanciers devises après chaque opération impliquant un mouvement de monnaies étrangères [Cfr III - 3.3.]

Le support de fonctionnalités bancaires complexes est donc l'un des principes que nous recherchons.

5.6. L'unicité des données

Enfin, l'aspect profondément centralisé du système et le fait d'obliger les programmeurs à se servir de certains outils montrent que l'unicité des données et des procédures est, elle aussi, un principe sous-jacent de ICE.

6. LES CONSEQUENCES SUR LA PRODUCTION DE LOGICIELS

Nous venons de mettre en lumière six principes généraux qui se retrouvent sans cesse dans l'environnement ICE :

1) l'utilisation d'une boîtes à outils,
2) la dissimulation d'information par le biais d'interfaces,
3) l'utilisation d'un langage de haut niveau,
4) la standardisation systématique,
5) le support de fonctionnalités bancaires de haut niveau,
6) l'unicité des données et procédures.

Tâchons maintenant de découvrir quelles peuvent être les conséquences du respect de ces principes sur la production de logiciels.

6.1. La hiérarchisation du système

Les deux premiers principes, l'utilisation systèmatique de boîtes à outils et l'utilisation systématique d'interface ont pour conséquence la structuration en couches : depuis la couche système jusqu'à la couche application. Chaque couche utilise les services de la couche inférieure.

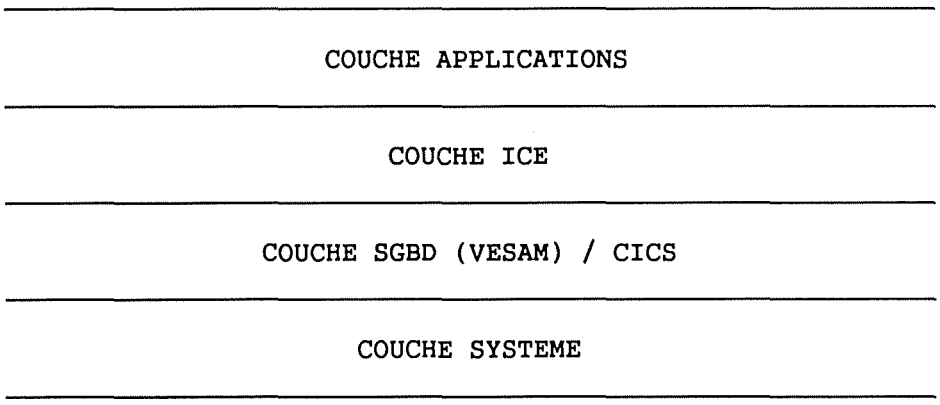


Fig. III - 6

Cette hiérarchie est basée sur la structuration du système en niveaux distincts grâce à la relation "utilise".

La définition formelle d'une hiérarchie "utilise" est donc la suivante :

Tout module de niveau 1 n'utilise aucun autre module. Et, si un module "A" utilise un module "B", alors "B" est un module de niveau inférieur au niveau du module "A".

Les remarques concernant la puissance de la relation "utilise" [Cfr III - 5.1.] sont ici aussi d'application.

6.2. La modularité des composants

Au sein même d'une couche, on définit différents composants appelés modules. Chaque module possède différents attributs : ses spécifications, son algorithme, ses performances, etc. De plus, il existe des relations entre ces modules. Nous trouvons, dans un texte de Gouthier et Pont [GT70], une définition de la philosophie de la programmation modulaire :

"A well-defined segmentation of the project effort ensures system modularity. Each task forms a separate, distinct program module. At implementation time each module and its inputs and outputs are well-defined, there is no confusion in the intended interface with other system modules. At checkout time the integrity of the module is tested independently; there are few scheduling problems in synchronizing the completion of several tasks before checkout can begin. Finally, the system is maintained in modular fashion; system errors and deficiencies can be traced to specific system modules, thus limiting the scope of detailed error searching..." [GT70].

Une autre définition proposée par Parnas est :

"... In this context 'module' is considered to be a responsibility assignment rather than a sub-program. The 'modularization' includes the design decisions which must be made before the work on independent modules can begin..." [PR72].

Parnas estime que le principal critère de décomposition en modules est celui de la dissimulation de l'information [Cfr III - 5.2.].

"Every module (...) is characterized by its knowledge of a design decision which it hides from all other. Its interface or definition was chosen to reveal as little as possible about its inner working ..." [PR72].

"We have tried to demonstrate by these examples that it is always incorrect to begin the decomposition of a system into modules on the basis of a flowchart. We propose instead that one begins with a list of difficult design decisions or design decisions which are likely to change. Each module is then designed to hide such decision from others. Since, in most cases, design decisions transcend time of execution, modules will not correspond to step in processing. To achieve an efficient implementation, we must abandon the assumption that a module is one or more sub-routines, and instead allow subroutines and programs to be assembled collection of code from various modules" [PR72].

Or, nous savons que chacune des interfaces est justement basée sur ce principe. De plus, elles réalisent toujours des services d'un même type. Nous avons déjà vu par exemple, que ICEFIC est spécialisée dans les problèmes d'accès aux données. Les modules-interfaces possèdent donc une forte cohésion procédurale et informationnelle, c'est-à-dire que tout module réalise des fonctions similaires sur base d'informations elles aussi similaires.

Nous venons donc de montrer que les différents composants de ICE possèdent les propriétés d'une architecture modulaire. En effet, chaque module possède :

- une forte capacité à cacher de l'information,
- un faible degré de couplage,
- une forte cohésion procédurale et informationnelle.

Ce sont là les critères mis en évidence par A. van Lamsweerde pour qu'une architecture soit modulaire.

6.3. La réutilisabilité des composants

Sommerville estime que "la réutilisation de programmes existants doit être encouragée autant que possible car cela diminue la quantité de codes à écrire, à tester et à documenter" [SM82].

La réutilisabilité est l'un des principes les plus importants de ICE. Elle est assurée par l'utilisation systématique des outils de la boîte à outils. Ils font un minimum d'hypothèses les uns sur les autres si leur paramétrisation est maximale.

Prenons par exemple ICEFIC, l'outil d'accès aux données et montrons qu'il est effectivement réutilisable et pourquoi.

La syntaxe nécessaire pour obtenir les services de ICEFIC est la suivante :

CALL 'ICEFIC' USING <OPERATION> <FICHIER> <PARAMETRES>

ICEFIC ne requiert aucune condition à son utilisation. Pas de fichier à ouvrir ni à fermer, pas de déclarations. Aucune hypothèse n'est nécessaire à son bon fonctionnement.

D'autre part, l'outil "ICEFIC" possède différents paramètres :

- La zone <OPERATION> contient le code de l'opération que le programmeur désire effectuer sur un fichier.

- La zone <FICHIER> cache de nombreux sous-paramètres : le nom dudit fichier, les codes d'erreur, la clef de l'enregistrement concerné etc.

- Enfin, il est possible que l'opération nécessite encore d'autres paramètres, tels les codes d'identification, qui sont alors placés dans la zone <PARAMETRES>.

On le voit, les outils de ICE vérifient les conditions de réutilisabilité. Que les applications élaborées grâce à ICE soient elles aussi réutilisables est une autre affaire. Ces dernières ne le seront que si elles aussi respectent les conditions du minimum d'hypothèse et du maximum de paramétrisation. Cela relève plus de la discipline personnelle des programmeurs que de la responsabilité de ICE.

6.4. La portabilité des applications

L'importance de la portabilité des applications est depuis longtemps connue :

"The rate of change of computer hardware technology is such that computing machinery becomes obsolete long before the programs which execute on these machines. It is therefore very important that programs should be written in such a manner that they may be implemented under more than one computer/operating system configuration" [SM82].

Sommerville estime que le type d'environnement basé sur le principe de la boîte à outils, ou si l'on préfère le principe des librairies, diminue le nombre de choses que le programmeur doit savoir et augmente la portabilité des applications :

"The programmer need not to learn the detail of several different systems. He is also preserved from disruptions caused by changes in the target hardware [SO82]".

Il signale aussi qu'il existe deux types de boîte à outils ou librairies :

1) Les librairies standards associées à une application particulière. Ces libraires possèdent une interface standard.

2) les librairies d'installation qui constituent un ensemble de routines proposées par les utilisateurs dans une installation particulière.

Il pense aussi que si l'utilisation des librairies de second type risque de diminuer la portablilité des applications, celle des librairies standards ne posent que peu de problèmes de portabilité.

Par son aspect de librairie standard, dans laquelle le programmeur vient puiser les outils nécessaires, ICE renforce la portabilité des applications. Si demain la Belgolaise décide de changer de SGBD [16], les applications ne changent pas. Ce qui doit éventuellement être modifié, ce sont les primitives ICE. Elles assurent donc la portabilité des programmes d'application vis-à-vis des logiciels de base et du matériel.
Par exemple, dans une autre société qui utilise aussi ICE, le remplacement de "VESAM" par "DL1" [26] n'a nécessité que 2 semaines de modification sur le noyau ICE et uniquement dans le module ICEFIC.

6.5. La diminution des coûts de maintenance

Plus un programme est structuré, documenté et indépendant des autres programmes, plus il est facile de le maintenir, c'est-à-dire de le corriger, de le modifier, ou encore de l'améliorer. ICE ne génère pas automatiquement la documentation mais la réutilisation des outils diminue d'autant les besoins en documentation.

L'utilisation obligatoire des outils de ICE est, elle aussi, un élément renforçant la lisibilité et donc la "maintenabilité" des programmes. Les programmeurs utilisent tous les mêmes primitives ICE. Pour tous, elles ont les mêmes fonctionnalités, les textes sont donc plus compréhensibles.

De plus, un programme ICE est toujours structuré de la même façon [Cfr III - 4]. Il est toujours divisé en un même nombre de blocs, chaque bloc ayant toujours la même fonction. Une personne étrangère à une application particulière pourra donc plus facilement reprendre le travail d'un collègue.

Une preuve empirique du fait que les coûts de maintenance sont diminués par l'utilisation systématique de ICE est, que dans d'autres organismes comparables à la Belgolaise (mêmes activités, même taille, même matériel), ces coûts de maintenance sont souvent deux fois plus importants. Rapplelons les travaux de Benington [BG56] et Boehm [BM84] qui ont montré que la maintenance représentait en moyenne 67 % des coûts des applications. A la Belgolaise, la maintenance intervient seulement pour 30 % des coûts dans le cycle de vie des logiciels [BL88].

6.6. Les standardisations à plusieurs niveaux

L'utilisation obligatoire de certains outils (ICEFIC [30]) et la réalisation implicite de certaines fonctions (sécurité, interface homme-machine) ont pour conséquence immédiate la standardisation. Cette standardisation se marque à deux niveaux :

1) au niveau de la conception, les applications sont toutes écrites dans le même langage, possèdent la même structure et utilisent les même outils;

2) au niveau des utilisateurs finaux, l'interface des applications est, elle aussi standardisée.

Cette particularité induit une conséquence importante : la mobilité du personnel de développement. Cette standardisation permet à un informaticien de passer facilement d'un projet à un autre. L'effort qu'il doit réaliser porte uniquement sur les concepts fonctionnels de l'application et non pas sur les outils ou les méthodes qui sont identiques dans toute la Banque.

6.7. La simplification de la programmation

La disponibilité de primitives puissantes, notamment au niveau de l'interface homme-machine, la standardisation des applications et la réalisation de niveaux dissimulant des informations, ont diminué la complexité de la programmation.

Par exemple, sur les gros système IBM, l'affichage d'écrans en COBOL doit passer par l'intermédiaire de fonctions CICS [25]. La programmation en CICS est très délicate et souvent inconnue des programmeurs débutants.

Les primitives ICE permettent de créer simplement des applications qui utilisent de nombreuses et complexes fonctions de CICS. Néanmoins, le programmeur ICE ne doit rien connaître de ces fonctions CICS. La programmation est donc grandement simplifiée par l'utilisation des primitives ICE.

6.8. La fiabilité des applications

Validité et robustesse sont les deux pilliers de la fiabilité. Si un programme est conforme aux spécifications de l'analyse fonctionnelle, s'il est complet, qu'il ne boucle pas, s'il respecte les contraintes d'intégrité des bases de données, alors on peut affirmer qu'il est valide.

Si ICE ne garantit pas la conformité aux spécifications, en revanche, il augmente le respect des contraintes d'intégrité des bases de données. Par exemple, si une transaction est abandonnée, le mécanisme de "ROLL-BACK" qu'il déclenche automatiquement diminue la probabilité d'une incohérence au niveau des données. De même, il peut, empêcher qu'un programme ne boucle. Il faut pour cela initialiser dans le répertoire des programmes une variable à une valeur correspondant au nombre maximum de transactions qu'il doit exécuter. Si le programme boucle, un module particulier de ICE s'en rend compte et met fin au programme.

6.9. L'optimalisation des ressources

Une conséquence très importante de l'utilisation de ICE est l'optimalisation de l'espace mémoire.

Anciennement, les programmes étaient compilés individuellement. Si deux d'entre-eux étaient exécutés "simultanément", le système chargeait brutalement l'ensemble des deux programmes en mémoire centrale. S'ils partageaient des parties de codes similaires, il y avait perte de place par redondance.

Les primitives ICE ont été écrites de manière complètement réentrante, c'est-à-dire que plusieurs programmes peuvent les utiliser "simultanément". Deux programmes qui utilisent chacun les services de "ICEFIC" et qui se voient exécutés en même temps sont eux aussi chargés en mémoire, mais le module "ICEFIC" n'est lui présent qu'une seule fois dans la mémoire du système.

L'ensemble du noyau ICE ne consomme que 30 Ko. On s'accordera à dire qu'il y a bien optimisation d'espace mémoire.

Mais l'utilisation de ICE induit aussi une amélioration des performances pour plusieurs raisons.

La première est que le noyau ICE a été réécrit en ASSEMBLEUR qui, on le sait, est nettement plus puissant que COBOL. L'ensemble des services auxquels fait appel le programmeur dans ses applications sont donc effectués plus vite que s'il avait dû les effectuer en COBOL standard.

D'autre part, du fait de l'optimisation de l'espace mémoire, le système doit moins souvent procéder à des accès disques et donc les programmes s'exécutent plus rapidement.

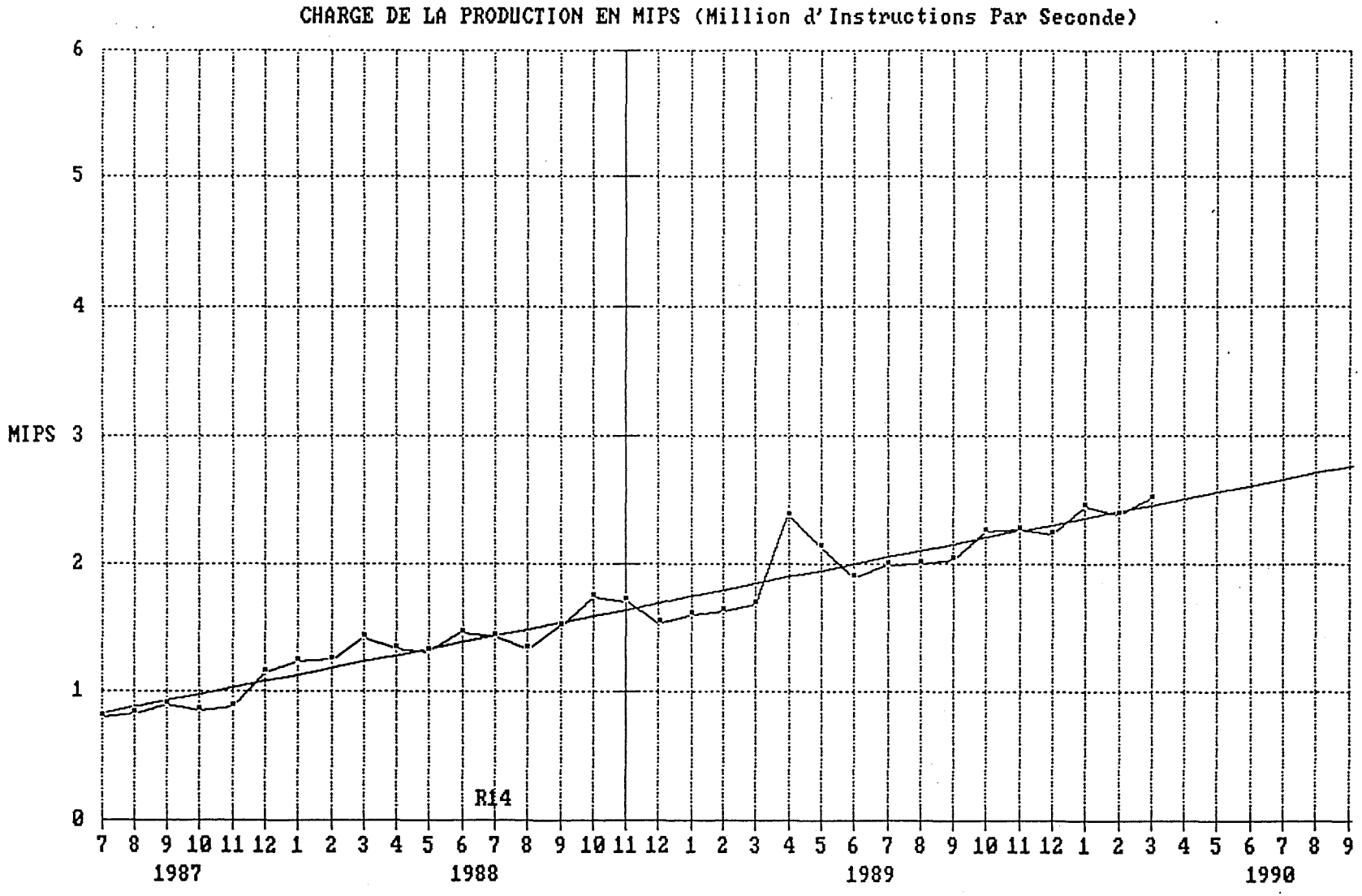
Le graphique de la figure III - 4 montre la charge en MIPS du mainframe. Chaque creux correspond au remplacement d'une application par son alter ego écrite en ICE. Il y a donc clairement économie de ressource CPU [32] par utilisation de ICE.

7. L'ARCHITECTURE ACTUELLE DU SYSTEME

L'architecture et l'implémentation de l'environnement sont fonction de la machine physique qui doit exécuter les applications. Par exemple, l'environnement ICE à la Belgolaise est supporté par deux outils principaux :

Un premier outil est un précompilateur qui traduit en COBOL le langage ICE dont nous venons de parler.

Un second outil est un moniteur qui assure le séquencement et le déroulement correct des transactions et des applications qui s'exécutent.


CHARGE DE LA PRODUCTION EN MIPS (Million d'Instructions Par Seconde)
MIPS
0
1
2
3
4
5
6
7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9
1987
1988
1989
1990
R14

- Fig. III.4. -

Cette dichotomie précompilateur-moniteur est due à une particularité du système d'exploitation de l'IBM. Sur ce système, lorsqu'un programme est momentanément interrompu (par exemple pour laisser le temps à un utilisateur final d'introduire ses données), l'exécution redémarre à la première ligne du code et non pas à la ligne qui suit l'endroit de l'interruption.

7.1. Les interfaces constitutives du noyau ICE

Le principe de base de ICE est celui de la boîte à outils [Cfr III-5.1.]. Ces outils forment le noyau du système. Ce noyau stratégique, appelé NOYAU ICE, assure différentes fonctions. Les fonctions du même type ont été regroupées en module ou "interface". Nous allons examiner maintenant de plus près chacune de ces interfaces. Mais leur étude systématique dépasse le cadre de ce mémoire. Le but de cette section est de présenter les principales fonctions de chacune des interfaces constitutives du noyau ICE.

a. L'interface d'accès aux fichiers (ICEFIC)

L'interface d'accès aux fichiers ICEFIC réalise de façon systématique et impérative l'ensemble des accès aux données [Cfr III - 3.1.]. Les applications n'ont qu'un unique moyen d'obtenir des informations. Elles doivent utiliser les services rendus par ICEFIC. C'est pourquoi, on considère que ICEFIC constitue le SGBD de ICE.

On peut aussi la considérer comme une interface entre les applications et les informations. La représentation physique des données est complètement transparente pour le programmeur. Que les informations soient sur bande ou sur disque, dans un fichier séquentiel ou indexé, n'apparaît pas dans le texte des applications. Celles-ci sont, du point de vue des requêtes sur les données, entièrement écrites en terme des primitives ICE.

b. L'interface de gestion des transactions (ICEMTX)

L'interface ICEMTX aussi appelé moniteur ICE, assure la gestion des transactions en "temps réel". Elle permet le séquencement correct des paragraphes des applications [Cfr III - 4.2.], la gestion des messages d'erreurs et celle des écrans.

Les applications sont clairement structurées en transactions logiques. L'enchaînement entre ces blocs logiques est entièrement géré à l'exécution par ICEMTX. Il s'occupe aussi de l'envoi, de la réception et de la séquence des écrans des transactions. Il facilite aussi la gestion dynamique du curseur et des attributs des champs des écrans (double brillance, couleurs, etc..).

ICEMTX garantit aussi l'intégrité des bases de données; si une transaction est abandonnée, grâce à un mécanisme de ROLL-BACK, il rétablit les bases données dans l'état dans lequel elles étaient avant la transaction. ICE agit de la même façon lors de l'abandon du programme, ne tenant compte physiquement que des transactions qui ont été jusqu'à leur terme normal, c'est-à-dire l'exécution correcte du paragraphe "ICETTR".

Les applications sont gérées de manière "pseudo-conversationnelle". Une particularité du système d'exploitation fait que les programmes ne sont actifs en mémoire qu'entre la réception de l'écran précédent et l'envoi de l'écran suivant. Ce procédé implique en principe la perte des zones de mémoire du programme. ICEMTX s'occupe de conserver ces zones mémoire.

La puissance de ICEMTX pour la souplesse d'utilisation des programmes est due également à des "touches fonction" préprogrammées qui permettent à tout moment de remonter la séquence de traitement écran par écran, de revenir directement au premier écran de la chaîne, ou de quitter immédiatement le programme.

ICEMTX gère aussi automatiquement l'envoi de message à l'écran. A chaque message est associé un niveau de sévérité choisi parmi quatre (Information, Warning, Error, Abend). ICEMTX prend les mesures correspondant au niveau de sévérité du message. Il est important de comprendre que la gestion de ces messages et de leurs conséquences est dynamique.

Par exemple, il existe une primitive ICE à laquelle le programmeur doit seulement fournir comme paramètre le numéro du message qu'il désire faire afficher. L'interface ICEMTX se charge d'aller lire dans un fichier le texte et le niveau de sévérité du message.

ICEMTX affiche le texte du message et, sur base de son niveau de sévérité, soit il :

- poursuit l'application (INFORMATION),
- exige un RETURN avant de poursuivre l'application (WARNING),
- exige une modification du dernier écran (ERROR),
- met fin à l'exécution du programme (ABEND).

c. L'interface de gestion du COBOL étendu (ICESBR)

L'interface ICESBR est l'interface de ICE qui assure la gestion du COBOL étendu. Grâce à elle, il est possible de définir des types de données particuliers.

Par exemple, le type de données "DATE". Il existe un format interne et un format externe du type "DATE". Le format externe est réservé à l'édition de la zone de données par exemple :

```
Format interne : SSAAMMJJ {PIC 9(8)}
Format externe : JJ / MM / SSAA
```

et l'instruction :

```
MOVE VAR_INT TO VAR_EXT [DATE]
```

signifie qu'il faut assigner la valeur de "VAR_INT" à "VAR_EXT" et que cette dernière est une variable d'édition de type "DATE". Les routines de ICESBR assurent automatiquement la conversion du format interne de "VAR_INT" en format d'édition.

De même, il est possible de définir des formats interne et externe du type "COMPTE". Lors de toute modification d'une variable de ce type, les primitives ICESBR vérifient la cohérence du numéro. Si les deux digits de contrôler ne sont pas conformes (digit = numéro_compte mod 97), un message d'erreur apparaît à l'écran et une correction est exigée.

Le programmeur n'a pas à gérer toutes ces opérations. Il se contente de demander la saisie d'un numéro de compte. Les vérifications et l'envoi de messages sont à charge de ICESBR.

d. L'interface de gestion des tâches background (ICEBG)

L'interface de gestion des tâches d'arrière-plan, plus communément appelée "moniteur des tâches background" permet de lancer à partir d'un terminal, des transactions s'exécutant sans écran. Elle effectue des lots de mises à jour en différé. Le terminal peut alors de nouveau être utilisé pour réaliser des travaux interactifs. ICE se charge du bon déroulement des tâches d'arrière-plan.

Les entrées et sorties sont stockées dans différents fichiers de même qu'un ensemble de statuts informent l'utilisateur du bon déroulement des opérations. Eventuellement, une évolution erronée peut déclencher un redémarrage de cette tâche.

e. L'interface d'impression (ICESPOOL)

Le "spool" ICE permet aux programmes d'application d'écrire, de conserver, de consulter et d'imprimer des listes sur des imprimantes. L'interface ICESPOOL permet aux programmeurs de réaliser ces fonctions d'une manière standard quels que soient le type et la localisation de l'imprimante. Pour ce faire, tout état imprimé doit avoir été défini au système dans un répertoire des listes. Ce répertoire a comme fonction première de définir, à l'aide de noms symboliques appelés "DESTINATION", sur quels périphériques physiques et en combien d'exemplaires seront imprimés les différents états.

Ce module fournit d'autres spécifications typiquement bancaires. Par exemple le concept "D'ORIGINAL". Lors de la déclaration d'une liste, il faut spécifier si le concept d'original s'applique ou non à cette liste. Si oui, un et un seul exemplaire sera imprimé sans autre commentaire. Tous les autres exemplaires porteront la mention "DUPLICATA". Cette notion d'original et de duplicata est très importante pour les listes servant de base à des opérations, afin d'éviter que la même opération ne soit exécutée plusieurs fois.

Enfin, il existe aussi une fonction de redémarrage automatique, c'est-à-dire que toute liste dont l'impression a été demandée sera imprimée, même après un arrêt du système.

En résumé, les fonctions "DESTINATION", "NOMBRE D'EXEMPLAIRES", "TYPE DE FORMULAIRE", etc ... sont prises en charge de manière paramétrée. En cas de changement, les applications n'ont pas à être modifiées, les adaptations sont assurées non par l'équipe de développement mais par l'équipe d'exploitation.

f. L'interface d'échange de données entre programmes (ICESAVE)

L'échange de données entre programmes est confiée à un module "ICESAVE". Sa fonction est assez semblable à celle des PIPE [19] sous UNIX. ICESAVE permet la définition de chaînes de programmes. Les données peuvent être sauvées puis récupérées dans n'importe quel ordre par un quelconque programme de la chaîne. Elles sont stockées dans une mémoire temporaire à accès rapide et sont détruites automatiquement lorsque le système arrive au bout de la chaîne.

g. L'interface "TP dégradé"

Cette interface a pour fonction de lire des données sur un fichier plutôt que de les lire à partir d'un terminal. C'est ce que l'on nomme communément la redirection des entrées et des sorties.

7.2. Le précompilateur

Le rôle du précompilateur ICE est de concevoir un programme COBOL compilable, à partir du texte de l'application rédigée dans le formalisme que nous avons exposé [Cfr III - 4.].

C'est la tâche du précompilateur que d'intercaler les instructions COBOL nécessaires qui feront que le programme, lors de son exécution, aura bien le comportement souhaité.

De plus, le précompilateur, se charge de rajouter toutes les instructions nécessaires pour assurer les multiples fonctions implicites de ICE (sécurité, interface homme-machine, ...).

Le précompilateur génère un programme COBOL. Ces sources COBOL sont elles-mêmes passées au compilateur COBOL et au LINKER afin de produire des codes exécutables.

En annexe, le lecteur trouvera le résultat de la précompilation de l'application présentée à la figure III - 4.a.

7.3. Les outils complémentaires

Nous avons déjà à plusieurs reprises parlé de différents outils, entre autres les différents répertoires. Il s'agit en réalité de programmes qui sont eux aussi disponibles dans l'environnement ICE. Ils rendent constamment de nombreux services aux développeurs.

Il existe actuellement dans l'environnement ICE de nombreux répertoires :

- fichiers,
- programmes,
- utilisateurs,
- listes,
- jobs,
- process,
- données,
- ...

Par exemple, le répertoire des programmes est indispensable pour assurer la sécurité. Ce répertoire crée aussi une association entre un fichier physique mémorisé sur un volume et un nom logique. Dans ses applications, le programmeur fait référence au nom logique; ICE, par l'intermédiaire de son répertoire des fichiers, fait le lien avec un fichier physique.

Il est donc possible à l'aide de cet utilitaire de modifier le fichier sur lequel travaille une application sans rien changer dans cette dernière.

Un autre outil appréciable de l'environnement ICE est le répertoire des données. Il donne de précieux renseignements sur les caractéristiques des données que manipule ICE. Format, taille, nombre, etc...

Sommerville signale à ce propos que : "A data dictionnary is a document which provides details of each and every entity relevant to the system being described...Data dictionaries are an important part of the documentation in some application areas, particularly those involving database management system [SM82]".

8. EVALUATION GLOBALE DE L'ENVIRONNEMENT ICE

En conclusion de ce chapitre de présentation de ICE, nous proposons de synthétiser et d'évaluer les différents aspects que nous avons abordés.

L'idée fondamentale de ICE est de créer un outil qui permette de diminuer les coûts de production et de maintenance des logiciels. Cet outil prend la forme d'un environnement logiciel. Des exposés généraux concernant les outils d'aide au développement de logiciels se trouvent dans [WG79], [RF80] et [SG81]. Cependant, signalons directement que selon A. van Lamsweerde :

"Les bénéfices d'un environnement logiciel 'évolué', en terme d'accroissement de la productivité et de la qualité du produit final, peuvent être substantiels. Les analystes/programmeurs peuvent d'avantage se concentrer sur les aspects difficiles et critiques de leur travail ... De tels produits permettent d'améliorer la communication et la coordination entre analystes/programmeurs, et peuvent constituer une documentation plus systématique en vue d'une maintenance ultérieure..."[VLWD].

Mais ICE constitue-t-il un bon environnement logiciel ? Pour répondre à cette question, il faudrait connaître les attributs qui devraient caractériser un environnement logiciel pour qu'il soit productif et utilisable à grande échelle.

A. van Lamsweerde [VLWD] propose une liste de cinq attributs :

1) intégration d'outils : l'utilisateur devrait avoir accès à un ensemble cohérent d'outils compatibles, complémentaires et à forte cohésion interne;

2) généralité, souplesse et individualisation possibles à des besoins particuliers;

3) couverture de la totalité du cycle de vie;

4) interface agréable à l'utilisateur;

5) Standardisation et transférabilité.

Au vu des sections précédentes, les points 1) (intégration d'outils) et 5) (standardisation et transférabilité) semblent acquis.

En effet, nous avons vu que la réutilisation d'outils était l'un des principes fondamentaux de ICE. La standardisation est elle aussi un fondement et une conséquence de l'utilisation de ICE.

Par contre, concernant le point 2), ICE n'est pas vraiment général puisqu'il est uniquement dédié à la réalisation d'applications transactionnelles.

Son principal défaut réside dans le fait qu'il ne couvre pas tout le cycle de vie des logiciels. En effet, ICE n'aide que très peu l'informaticien durant sa tâche de conception générale.

Si ICE facilite la programmation, il n'est d'aucune aide du point de vue de l'analyse.

Le Professeur Van Lamsweerde signale d'ailleurs qu'un "environnement ne sera, à notre avis, jamais une panacée; rien ne peut combler l'absence de méthodes sérieuses de développement de logiciels. En attendant, des outils (...) pourront rendre la vie de l'informaticien plus agréable et plus productive" [VLWD].

Selon le département informatique de la Banque, l'environnement ICE permettrait d'accroître la productivité des informaticiens de 25 %, et la maintenance à la Belgolaise n'intervient que pour 30 % dans l'ensemble des coûts des logiciels. Ces pourcentages ont été établis sur base de statistiques extraites des budgets successifs du département [BL88].

En conclusion de ce qui précède, on peut affirmer que le caractère réutilisable des composants, l'amélioration de la maintenance, de la portabilité , de la convivialité, de la standardisation, de la fiablilité et des performances des applications induites par l'utilisation de ICE permettent de diminuer les coûts de production des logiciels. ICE augmente donc la productivité des informaticiens. Il atteint là son objectif principal.

CHAPITRE IV : LE PROJET DE DECENTRALISATION

CHAPITRE IV : Le projet de décentralisation

Le chapitre I présente la Belgolaise.

Le chapitre II détaille son système d'information.

Le chapitre III étudie en détail la principale caractéristique de son informatique, à savoir l'environnement de développement ICE.

Le chapitre IV, présente pour sa part, un projet dont le but est de produire une version PC de l'environnement ICE ou plus exactement de l'adapter à un réseau d'ordinateurs personnels.

Ce chapitre s'articule selon 5 sections :

La première où nous définissons plus précisément les objectifs de ce projet de décentralisation et l'idée qui le sous-tend.

La deuxième où nous présentons diverses solutions concurrentes qui ont été envisagées pour réaliser ce projet générique de décentralisation.

La troisième expose en détail l'élaboration du projet spécifique dont le but est de produire une version PC de ICE à partir de FBSS.

La quatrième contient les conclusions de ce chapitre.

1. L'OBJECTIF DU PROJET

A la fin des années 80, l'informatique connaît d'énormes bouleversements. Les micros font leur apparition dans l'entreprise. A cette époque et pour le seul Siège de Bruxelles, on compte déjà 50 PC. La puissance de ces machines individuelles engendre plusieurs conséquences dont la première est la diminution relative du prix de revient du MIPS [12]. Les PC peuvent alors être vus comme un produit de substitution des mainframes. La solution des problèmes soulevés par l'explosion des besoins en calcul passe peut-être par l'adjonction de nombreux micros à la machine centrale.

A ce sujet, un exemple peut-être très révélateur. A Londres, le Nixdorf vient d'être remplacé par un PC sous Unix. Le coût de la conversion est amorti en 3 ans par la disparition des frais de location et de maintenance des logiciels du système précédent.

Malheureusement, cette solution possède, elle aussi, de sérieux défauts. Les bases de données en se multipliant deviennent redondantes, voire incohérentes. On ne compte plus les fichiers perdus ou les disques détruits. Les gestionnaires doivent prendre des mesures organisationnelles contre cette anarchie grandissante induite par l'utilisation désordonnée des PC.

Parallèlement à l'apparition des ordinateurs personnels, le concept de réseau local [21] émerge dans le paysage informatique de la Belgolaise. Un Token Ring [15] relie différents PC et leur permet de dialoguer entre eux et avec le mainframe.

Enfin, la Belgolaise est satisfaite par l'environnement de développement qu'elle utilise sur le mainframe. Elle désirerait l'adapter aux nouveaux matériels (PC et TOKEN RING) qu'elle utilise de plus en plus.

La Belgolaise pourrait donc se doter dans les prochaines années d'un système d'information dont l'architecture globale serait la suivante : un système central, secondé par de puissants PC, le tout relié par un réseau local et intégré dans un environnement de programmation cohérent.

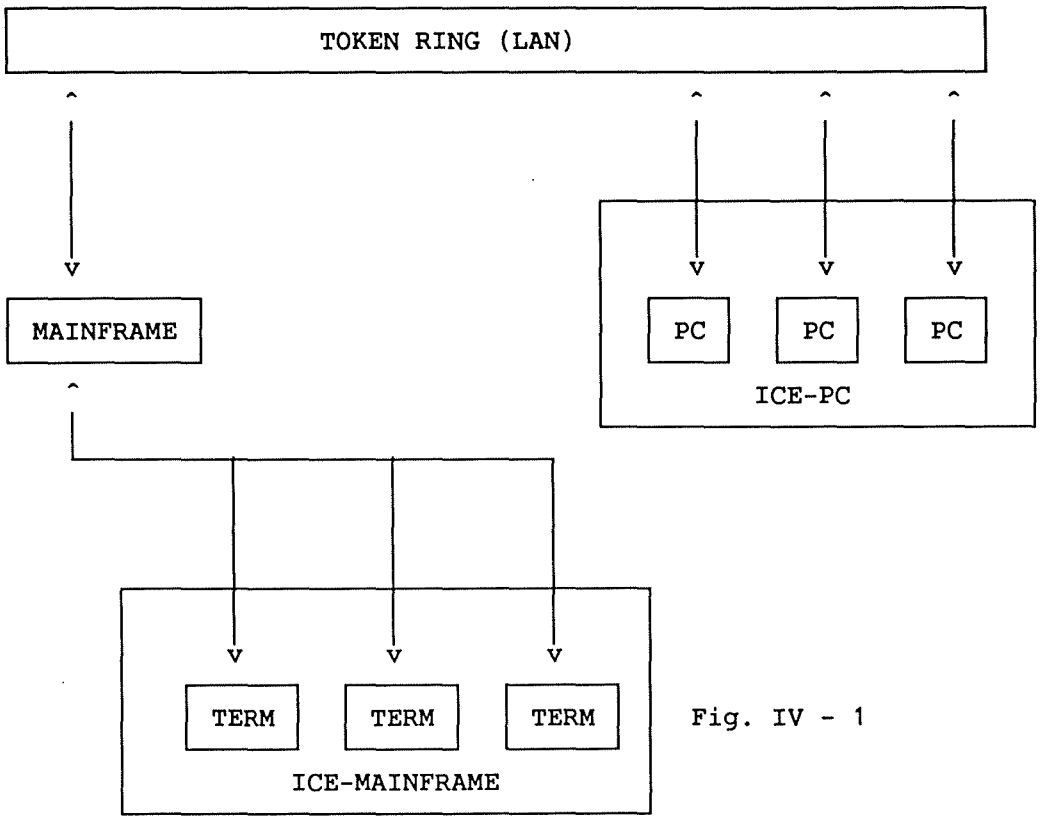


Fig. IV - 1

En résumé, la philosophie du projet est de décharger le mainframe au profit d'un réseau de PC qui bénéficierait des mêmes facilités de développement que le site central, c'est-à-dire de l'environnement ICE.

Il serait alors possible de créer des applications sur PC avec les mêmes méthodes et dans le même formalisme que sur le mainframe. De plus, il serait aussi possible d'exploiter ces applications sur le réseau de PC. Cela déchargerait le mainframe, éviterait son remplacement et enfin, augmenterait la disponibilité générale du système informatique global.

Loin de vouloir assurer la réalisation intégrale de cette décentralisation, le projet poursuit cependant un double objectif. D'une part, en évaluer la faisabilité et l'opportunité. D'autre part, en apprécier les grands choix de conception et en élaborer un premier prototype.

L'exposé qui précède présente un projet générique. Cependant il existe plusieurs façons de le réaliser. Les différentes solutions envisagées ont chacune donné naissance à un projet spécifique. Nous proposons dans les sections qui suivent de dresser l'inventaire de ces différents projets.

2. LES DIVERSES SOLUTIONS ENVISAGEES POUR REALISER LE PROJET GENERIQUE

2.1. L'achat d'un environnement pour PC

La première solution qui vient à l'esprit, la plus simple pour la Belgolaise, est d'acheter un environnement de programmation répondant aux caractéristiques de ICE.

Nous avons vu dans l'exposé du schéma directeur [Cfr II - 1.6.] qu'en règle générale, la banque préfère acheter les logiciels de base et développer les applications relevant de son activité particulière.

Le développement d'un atelier logiciel ne relève pas de l'activité particulière de la Banque. Cependant, comme pour le système central, la Belgolaise a préféré développer elle-même son propre environnement sur PC. Deux raisons essentielles ont motivé ce choix :

Primo, il n'existe actuellement sur le marché aucun standard. A notre connaissance, aucun éditeur de logiciels ne propose d'environnement de programmation similaire à ICE et qui fonctionne sur PC.

Secundo, l'achat d'un nouvel environnement de développement forcerait les analystes et programmeurs à adopter un nouveau "style" de programmation. Ce type de changement, le plus souvent mal vécu par les intéressés, augmente inutilement leur apprentissage.

Ces deux raisons ont poussé la Belgolaise à développer son propre environnement sur PC. Ce sont probablement les mêmes raisons qui l'ont poussée à développer ICE en site central. La Banque, après avoir acheté la licence du système à la SOGENAL, et après en avoir évalué avantages et désavantages, en a décidé la refonte afin de l'adapter plus complètement aux exigences de son informatique.

La Belgolaise préfère donc produire elle-même une version PC de ICE plutôt que d'acheter un environnement de programmation par trop différent de ICE.

2.2. Le développement d'une version PC de ICE

L'option prise par la Belgolaise a donc été de développer une version PC de ICE. Mais il existe encore diverses façons de procéder.

L'une d'elles, parmi les plus prometteuses, consiste à adapter DBASE [20] à ses besoins.

Une autre voie de réflexion consiste à développer le projet à partir de FBSS [4].

La solution DBASE est présentée à la section suivante et celle basée sur FBSS est abordée dans la dernière section de ce chapitre.

a. La solution DBASE IV

DBASE, est un système de gestion de base de données introduit depuis longtemps à la Belgolaise. Il est utilisé pour plusieurs applications PC.

Mais, DBASE est plus qu'un SGBD. C'est aussi un puissant langage de programmation parfaitement adapté aux applications du domaine bancaire. DBASE est depuis longtemps un standard du marché. Il possède aujourd'hui tant d'utilitaires et de possibilités qu'il constitue déjà presque à lui seul un environnement de développement. Le problème est de savoir s'il est suffisamment adaptable pour simuler l'environnement ICE.

Cette étude a été confiée à une équipe de deux programmeurs. La voie qu'ils ont ouverte semble assez prometteuse. Nous en proposons un rapide résumé.

Le but de cette initiative était double :

1) tester un nouveau logiciel appelé sans doute à devenir un standard.

2) déterminer si DBASE IV pouvait constituer un environnement standardisé du développement sur réseau de PC.

La principale difficulté de cette approche consiste à adapter le langage et la structure de DBASE. En effet, comme nous l'avons montré dans le chapitre précédent [Cfr III - 4], le langage ICE est un langage d'un niveau supérieur à COBOL qui en est la base. Il fallait donc examiner s'il était possible de casser la structure du langage DBASE pour la faire ressembler le plus possible à celle du langage ICE.

Il est possible aujourd'hui d'écrire des applications dans un langage ICE-DBASE qui ressemble assez au langage ICE original. Le lecteur intéressé peut se reporter en annexe, il trouvera le texte d'une application écrite dans ce langage.

Signalons que cette expérience a aussi permis d'établir une série de normes que les développeurs d'applications sur PC sont maintenant obligés de respecter. Ces normes facilitent une standardisation du développement sur micro.

Outre ces divers aspects, ICE-DBASE bénéficie de différents avantages. Notons entre autres que :

1) la gestion du réseau est assurée par DBASE,

2) DBASE comprend un compilateur d'application,

3) DBASE supporte SQL,

4) il existe de nombreux liens avec les logiciels standards utilisés à la Banque (MULTIPLAN, WP, CHART, LOTUS),

5) la portabilité vers OS/2 est garantie par l'éditeur du logiciel.

Cependant, dans la version actuelle de ce prototype, il n'existe aucun lien entre les PC fonctionnant sous ICE-DBASE et le mainframe. Les bases de données sont déchargées manuellement vers le micro. C'est-à-dire que les fichiers mainframe sont transférés dans un format PC grâce à un logiciel de communication spécialisé. Ensuite, ils sont copiés sur une ou plusieurs disquettes et enfin, déchargés sur le disque dur du PC concerné.

En résumé, la solution DBASE peut se représenter comme suit :

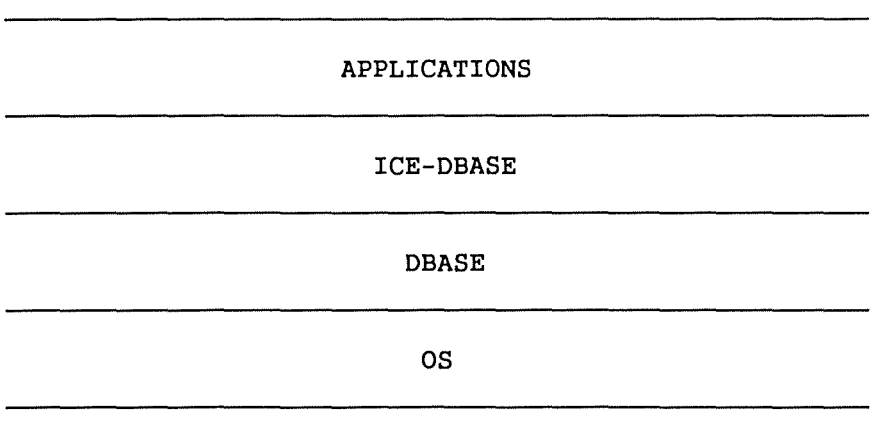


Fig. IV - 2

b. La solution FBSS

Nous avons vu à la section précédente que DBASE constitue une solution par laquelle il est possible d'élaborer la version PC de ICE. Cependant, d'autres logiciels peuvent eux aussi servir de point de départ à ce projet.

Un autre point de départ peut être un produit qu'IBM vient de commercialiser : FBSS [4]. C'est un logiciel d'assez bas niveau qui constitue une couche intermédiaire entre le système d'exploitation, le réseau et les applications.

La solution FBSS peut se représenter comme suit :

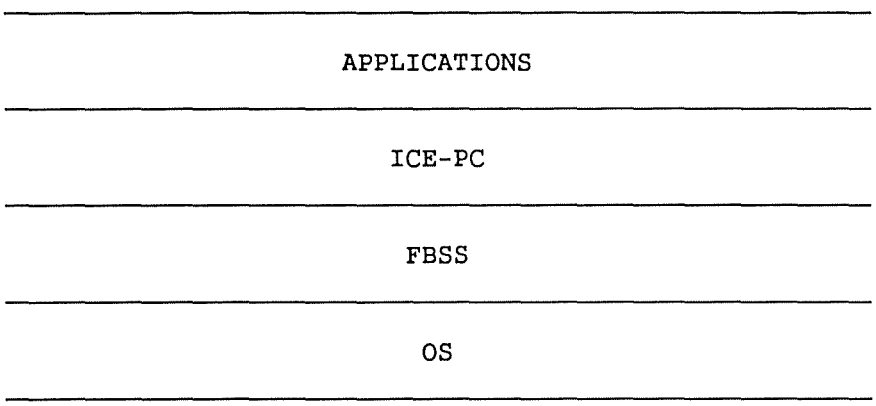


Fig. IV - 3

Le projet qui consiste à produire la version PC de ICE à partir de FBSS s'appelle ICE-PC. Dans le chapitre suivant nous en proposons la description complète.

3. LE PROJET ICE-PC BASE SUR FBSS

La première section présente un projet générique dont le but est d'adapter l'environnement ICE à un réseau de PC. Nous en connaissons maintenant l'objectif et nous avons brièvement décrit les deux projets concurrents qui tentent chacun de produire la version PC de ICE.

La réalisation du premier de ces projets, celui qui se fonde sur DBASE, a été confiée à une équipe de deux informaticiens de la Belgolaise. Nous laisserons de côté cette voie de recherche pour nous concentrer sur le second de ces projets.

Cette section présente donc les spécifications, la conception et l'implémentation d'un prototype d'une version PC de ICE basée sur FBSS.

Cette section s'articule logiquement autour de 3 sous-sections :

La première définit rigoureusement le cadre de ce projet ICE-PC.

La seconde détaille par raffinements successifs la conception de cette solution.

Enfin la troisième section expose le développement du prototype dérivé de ce qui précède.

3.1. Définition du cadre du projet

3.1.1. Définition du domaine et des contraintes

Les objectifs du logiciel que nous avons à développer peuvent être définis de la manière suivante : ICE-PC doit simuler le plus parfaitement possible les caractéristiques de la version fonctionnant sur l'IBM 4381 [Cfr III]. A la limite, les applications ICE écrites pour le mainframe devraient fonctionner sur les micros sans qu'aucune modification ne soit nécessaire. Il suffirait de prendre les programmes ICE du mainframe et de les compiler dans l'environnement ICE-PC pour qu'ils soient exécutables sur PC.

La principale contrainte que nous devons respecter tient à la nature du matériel sur lequel nous devons travailler et à la stratégie informatique de l'entreprise.

Nous savons que l'un des grands principes de la Banque est l'unicité des données. En conséquence, les bases de données, qui se trouvent déjà sur l'IBM, ne peuvent être déchargées sur le PC. De toute façon, les faibles capacités de stockage des PC interdisent la décentralisation des données. Ces dernières doivent donc impérativement rester sur le site central.

Il est donc nécessaire d'assurer une liaison fiable entre ce dernier et les PC. ICE-PC doit donc supporter un réseau. Or, nous l'avons déjà signalé dans le chapitre premier, la Banque possède un réseau reliant l'IBM et différents PC. Il s'agit du TOKEN RING [15].

En résumé, il nous est demandé de développer, dans le cadre d'un réseau de micros ordinateurs, un environnement de développement d'applications transactionnelles qui simule l'environnement ICE existant sur le mainframe.

3.1.2. Restriction par rapport à la version mainframe

L'objectif et les contraintes temporelles fixés pour la réalisation de ce projet dépassent clairement le cadre de ce mémoire. Il se limite donc au seul aspect accès aux données. Les autres fonctionnalités de ICE n'ont pas été adaptées à l'environnement PC.

Cependant, pour être plus complet, nous avons aussi réalisé un précompilateur [Cfr III - 7.2.] qui permet d'écrire des applications dans un formalisme similaire à celui utilisé sur le mainframe. [Cfr III - 4]

3.2. Conception de la solution

Nous avons défini dans les sections précédentes les objectifs et les contraintes du projet. Nous présentons maintenant la solution que nous avons élaborée. Cette présentation se compose de deux étapes :

primo, la conception globale du système;

secundo, la spécification des composants, c'est-à-dire la conception détaillée de la solution.

Durant ces deux phases de conception, nous utilisons une méthodologie dite "top-down". Des auteurs comme Wirth [WR71] et Dijkstra [DK68] considèrent en effet que cette approche est plus pertinente pour la conception d'ensemble des systèmes.

3.2.1. Conception globale de la solution

La solution globale comporte un aspect logiciel et un aspect matériel. Présentons d'abord ce dernier aspect de la solution.

a. Architecture matérielle

ICE-PC repose sur une configuration particulière d'éléments matériels :

1) le mainframe de la Belgolaise (IBM 4381),
2) les PC (IBM PS2 MOD30 et MOD50).

Ces différentes machines sont reliées entre elles par un LAN [21] de type TOKEN RING [15]. Une carte et un logiciel spécifiques sont rajoutés à chaque micro et en assurent l'interface avec le réseau.

L'un des PC est utilisé comme serveur de fichiers. L'autre est réservé aux développements et aux tests. Les échanges avec le mainframe sont réduits au minimum pour des raisons de sécurité.

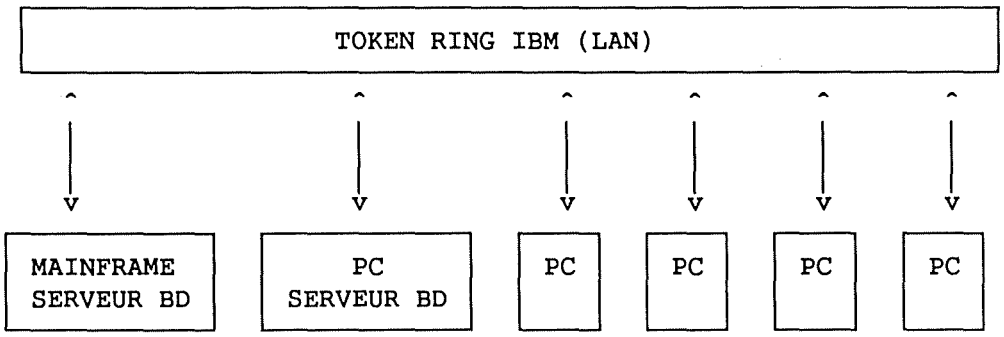


Fig. IV - 4

b. Architecture logicielle

ICE-PC repose aussi sur une configuration particulière de programmes. Cette architecture logicielle peut se représenter de façon hiérarchique :

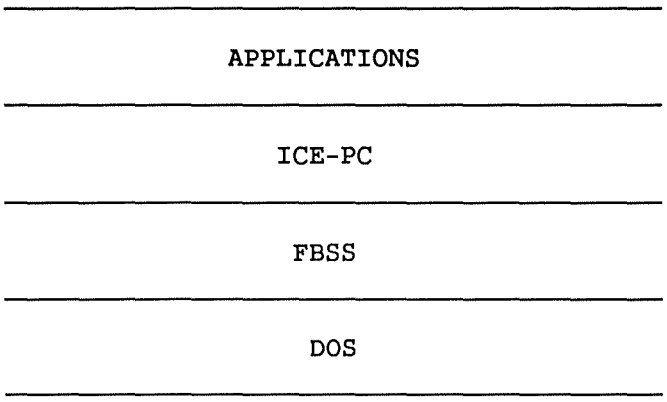


Fig. IV - 5

Les remarques sur la hiérarchisation et la relation "utilise" [Cfr III - 5.1.] s'appliquent à nouveau ici.

Il existe donc dans l'architecture de notre système, quatre couches logicielles superposées. Chacune utilise les services des couches inférieures et en propose de plus complexes.

Dans les sections suivantes, nous examinons en détail chacun de ces niveaux.

- Le niveau DOS

Le premier des niveaux de l'architecture logicielle est constitué par le système d'exploitation des PC. Il s'agit dans ce cas du DOS 3.3 de MICROSOFT.

En quelques mots, MS-DOS [5] utilise les primitives et les interruptions matérielles pour mener à bien toutes les tâches qui lui sont assignées, à savoir :

1) la gestion de la mémoire vive,
2) la gestion des entrées-sorties,
3) la gestion des mémoires de masses,
4) la gestion de nombreux périphériques.

Nous ne nous étendrons pas plus sur ce niveau, signalons seulement que c'est au niveau MS-DOS qu'il faut définir la carte d'interface avec le LAN.

- Le niveau FBSS

La seconde couche de l'architecture logicielle est assurée par FBSS [4]. Le "Financial Branch System Service" est un logiciel qu'IBM vient de commercialiser. C'est une extension du système d'exploitation de l'IBM PC (MS-DOS). Son rôle est de fournir différents services aux programmes d'application. Ces services facilitent la gestion des communications et du réseau local.

FBSS a été spécialement conçu pour faciliter la réalisation d'applications financières. Cependant, FBSS peut aussi constituer la base d'autres systèmes transactionnels. Il facilite notamment :

- le partage des ressources (fichier, imprimante, ...),
- l'interconnection des ordinateurs via un réseau local,
- les communications entre différents réseaux locaux,
- l'émulation d'un terminal mainframe.

Nous allons maintenant définir différents concepts de l'environnement FBSS.

Afin de mieux les comprendre supposons que des PC ayant des périphériques soient connectés entre eux via un réseau local.

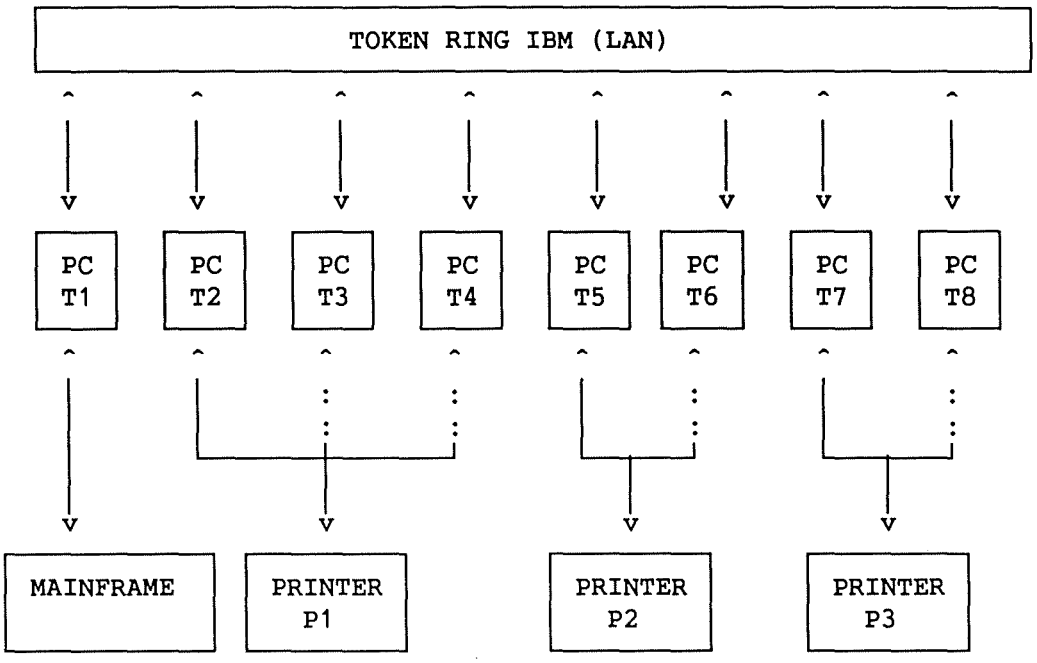


Fig. IV - 6

Le PC qui est physiquement connecté à une ressource (trait plein) en est le propriétaire. Pour la partager avec d'autres PC du réseau le propriétaire d'une ressource doit disposer d'un serveur.

Un serveur est un programme qui contrôle et gère l'accès à la ressource.

Il existe différents types de serveurs disponibles dans l'environnement FBSS :

1) Les serveurs de communication qui se chargent du dialogue inter-systèmes.

2) Les serveurs de fichiers qui stockent et partagent les données réparties.

3) Les serveurs de répertoires DOS qui simulent un disque dur pour les PC qui n'en sont pas équipés.

4) Les serveurs de périphériques financiers ("Financial I/O Device") qui impriment, affichent et saisissent des données.

Par exemple, le PC "T5" du graphique ci-dessus, est propriétaire de la ressource (l'imprimante "P2") et la partage avec l'ordinateur personnel "T6" via le LAN. "T5" doit posséder un serveur d'imprimante. De la même façon, un PC peut être propriétaire d'un fichier "F" et le partager avec les autres PC connectés au LAN. Le propriétaire du fichier partagé doit avoir un serveur de fichiers qui contrôle et gère l'accès à "F".

Un PC équipé d'un serveur et du périphérique correspondant installé de façon adéquate est appelé le propriétaire du serveur et par extension le serveur.

Tout programme d'application d'un PC du LAN peut obtenir un service d'un serveur. L'endroit où le serveur se trouve, dans le même PC ou sur un autre, est transparent pour l'application appelante.

Les machines qui doivent accéder à un serveur sont définies lors de la configuration de FBSS. Durant les opérations journalières il n'est donc pas nécessaire de savoir quel PC est propriétaire de la ressource. C'est au noyau FBSS [voir infra] de déterminer le chemin et de réaliser l'accès aux données. Tout se passe du point de vue de l'application comme si les données étaient sur la même machine que celle sur laquelle elle s'exécute. On utilise les termes de "connection logique" pour désigner ce mécanisme.

Le système FBSS est aussi ouvert aux serveurs non FBSS. L'organisation qui utilise FBSS peut, par exemple, développer un serveur qui gère une installation ou des services partagés particuliers.

FBSS est composé de différents programmes. Ceux-ci peuvent être regroupés en cinq groupes :

1) les primitives FBSS;

2) les applications internes activées par l'utilisateur, tel "l'interface opérateur", "l'émulateur 3270" et les outils de "TRACE" et "DEBUG";

3) les applications internes "Timer-activated", tel "l'émulateur d'imprimante 3287";

4) les programmes FBSS activés et contrôlés par commande DOS : le programme SVP, le "Host File Transfert", le "Share File Directory";

5) les utilitaires FBSS activés et contrôlés par commande DOS : le programme "FBSS Loader", les utilitaires d'installation et de configuration de FBSS.

En plus, d'autres programmes assurent certaines fonctions particulières.

Par exemple :

- Le "Supervisor-Router" est un programme résidant dans la mémoire centrale de chaque PC du système qui constitue une extension du DOS de chacune de ces machines.

  Ce programme FBSS, assure les fonctions du premier groupe de programmes identifiés plus haut. Il est responsable de :

  1) la réception des requêtes des programmes d'application par l'interface de programmation;

  2) le routage des requêtes vers serveurs appropriés;

  3) le renvoi des données à l'application appelante;

  4) la gestion des évènements asynchrones;

  5) la gestion des serveurs;

  6) l'activation des applications internes (par exemple l'émulateur 3270 lorsque l'utilisateur appuie sur la touche correspondante);

  7) la restitution du contrôle, à intervalle régulier, au gestionnaire temporel des applications internes;

- Le "LAN Session Services" est responsable de la conduite des communications entre les machines connectées au LAN. FBSS utilise les sessions LAN pour accomplir différentes tâches :

  1) le partage des ressources (imprimantes, fichiers, ...);

  2) l'exécution sur une machine d'un programme stocké sur une machine quelconque du réseau;

3) l'indépendance des programmes d'application face aux changements de configuration;

4) la transparence; l'utilisateur ne doit pas savoir quelle machine physique est propriétaire d'une ressource. Le système sait où se trouve les ressources et se charge de leur faire parvenir les requêtes.

En résumé, les fonctions qui nous intéressent à ce niveau sont celles qui facilitent l'accès aux données réparties sur le réseau. En effet, certaines primitives de FBSS permettent à un PC d'obtenir, en réponse à une requête, les données d'un fichier se trouvant sur un quelconque ordinateur du réseau.

Le problème majeur de FBSS est que le formalisme, ou si l'on préfère la syntaxe des requêtes, est extrêmement complexe. Pour utiliser FBSS, il faut remplir des zones prédéfinies à l'aide de code hexadécimaux. Cette dernière opération est extrêmement délicate et pour le moins peu conviviale. Ce problème est résolu par la couche logicielle supérieure que nous détaillons à la section suivante.

- Le niveau ICE-PC

La troisième couche de la hiérarchie est la couche ICE-PC qui permet aux applications de niveau supérieur de réaliser des primitives dans la syntaxe du langage ICE. Cette couche ICE-PC pourrait contenir chacune des interfaces constitutives du noyau de la version mainframe de ICE [Cfr III - 7]. Cependant, étant donné les restrictions formulées au point 3.1.2. de ce même chapitre, la couche ICE-PC n'est composée que du seul module ICEFIC.

Ce module ICEFIC prend en charge les requêtes d'accès aux données que lui adressent les applications. Ces dernières ne doivent connaître que la façon dont on formule des requêtes à ce module, c'est-à-dire la syntaxe ICEFIC.

La syntaxe ICEFIC est basée sur le mécanisme du "CALL COBOL". Ce mécanisme garantit la transparence. Les développeurs des applications ICE-PC n'ont rien à connaître de FBSS. Ce sont les programmes du niveau inférieur et dans ce cas, du module ICEFIC, qui utilisent les services de FBSS pour réaliser les requêtes via le réseau local. L'utilisation de FBSS est le secret du module ICEFIC et est inconnue aux applications.

Par contre, le module ICEFIC utilise intensément les services de FBSS. Il transforme les requêtes des applications en requêtes FBSS. Ensuite il renvoie aux applications les résultats reçus de FBSS. Ces résultats sont eux aussi traduits dans la syntaxe ICEFIC.

- Le niveau application

La couche supérieure de la hiérarchie est celle des applications. Celles-ci sont écrites dans un langage très proche du langage ICE de la version mainframe. Une fois écrites, les applications ICE-PC sont traduites par le précompilateur ICE-PC que nous avons développé à cette fin.

Ces applications ICE-PC possèdent deux caractéristiques syntaxiques.

1) Les applications ICE-PC peuvent contenir des requêtes d'accès aux données. Ces requêtes sont uniquement exprimées dans le formalisme de l'interface ICEFIC.

2) Les applications ICE-PC sont structurées dans le langage ICE : toute application contient donc les paragraphes descripteurs d'une transaction, à savoir : ICEIPR, ICEITR, ICE001-I, ICE001-T, ICE001-E, ..., ICETTR, ICETPR [Cfr III - 4].

3.2.2. Conception détaillée des composants de la solution

Nous venons d'exposer la conception globale de la solution, à savoir son architecture matérielle et logicielle. Nous abordons dans ce qui suit, la conception détaillée de la couche ICE-PC. Les deux autres couches, la couche inférieure (FBSS) et la couche supérieure (APPLICATION) ne sont pas de notre ressort.

Nous avons montré précédemment que les deux composants que nous avons développés dans notre solution sont le module ICEFIC et le précompilateur ICE-PC. Les deux points qui suivent sont respectivement consacrés à la conception détaillée de chacun de ces deux composants.

a. Conception de ICEFIC

Le module ICEFIC se charge de réaliser les requêtes d'accès aux données qui lui sont adressées. Ensuite, il fournit les résultats et les codes d'erreurs au programme appelant.

Notons qu'en réalité, lorsqu'un programme s'exécute, la couche ICEFIC a été intégrée à l'application. Dans la version mainframe de ICE, ceci n'est pas vrai; le module ICEFIC est physiquement toujours présent en mémoire. Tandis que sur PC, le module ICEFIC est compilé une fois pour toutes et est intégré au code de l'application par le LINKER. Seule la couche FBSS est encore présente physiquement et de façon indépendante dans la mémoire centrale du PC.

Cette différence est due au caractère multi-tâches et multi-utilisateurs du mainframe. Sur ce système, deux programmes peuvent s'exécuter "simultanément". Ces programmes peuvent avoir des parties identiques de codes. Par exemple, les primitives ICE sont suceptibles d'être redontantes dans chacun des programmes. L'idée, pour économiser de la place mémoire, est de les faire partager ce code commun. Il faut pour cela écrire ces primitives de manière totalement "réentrante". Ce mécanisme permet à un même code machine d'être exécuté "simultanément" sans duplication de ce code.

Le PC est une machine mono-utilisateur et mono-tâche. Le problème de la duplication du code ne se pose donc pas. C'est pourquoi, il n'est pas nécessaire de garantir le caractère réentrant des primitives ICE. Le code des primitives ICEFIC peut donc être directement inséré dans le code des applications.

Précisons maintenant quelles sont les fonctions de ce module ICEFIC. Une manière de définir les fonctions d'un module consiste à spécifier son interface.

La syntaxe de l'appel aux fonctions de ICEFIC est la suivante :

CALL 'ICEFIC' USING <FUNCTION>, <PARAM>, <FICH>

<FUNCTION> précise le type d'accès fichier que l'on désire réaliser. Par exemple, "RF" signifie que le programmeur désire réaliser une lecture du premier élément d'un fichier séquentiel.

Le programmeur doit pouvoir effectuer 8 fonctions :

1) lecture séquentielle du premier enregistrement : "RF" (Read First)
2) lecture séquentielle de l'enregistrement suivant : "RS" (Read Seq)

3) lecture indexée d'un enregistrement : "RI" (Read indexed)
4) écriture d'un enregistrement : "WR" (WRite)
5) modification d'un enregistrement : "RW" (ReWrite)
6) effacement d'un enregistrement : "DL" (DeLete)
7) début de la transaction : "BT" (Begin Transaction)
8) fin de la transaction : "ET" (End Transaction)

<PARAM> précise si l'utilisateur désire procéder à la lecture d'un enregistrement suivie ou non d'un verrouillage de ce dernier. Pour modifier ou effacer tout enregistrement, il doit avoir été préalablement verrouillé. Toute autre valeur de <PARAM> doit être interprétée comme une lecture sans verrouillage. Un enregistrement verrouillé par une application ne peut plus être modifié par une autre application. Enfin, un enregistrement verrouillé par une application est automatiquement déverrouillé si cette application procède à un nouveau verrouillage ou à un "End Transaction".

En résumé, <PARAM> spécifie si la lecture est suivie :

1) d'un blocage de l'enregistrement : "LO" (LOck)
2) d'aucun blocage : "NL" (Not Lock)

<FICH> contient les caractéristiques du fichier auquel on désire accéder et les données en elles-mêmes. Il se compose donc de deux sous-paramètres <ENTETE> et <ENREG>.

<ENTETE> contient toutes les caractéristiques concernant ce fichier, à savoir :

1) le nom logique du fichier [*].
2) le numéro de la clef.
3) le code d'erreur détecté par ICEFIC [**].
4) la longueur des données contenues dans <ENREG>.

[*] Le programmeur d'application fait référence à un nom logique de fichier. Au niveau de ICEFIC, il faut faire le lien entre ce nom logique et un nom réel du fichier existant du système. ICEFIC doit accéder à un fichier technique de l'environnement ICE-PC. Ce fichier technique est l'équivalent du répertoire des fichiers de l'environnement ICE sur le mainframe. Ce mécanisme permet de rendre les applications indépendantes du nom physique des fichiers.

[**] Les codes d'erreurs susceptibles d'être renvoyés par ICEFIC sont les suivants :

-0- la requête s'est déroulée correctement.

Les autres codes d'erreurs signalent que la requête ne s'est pas déroulée correctement :

-4- enregistrement non trouvé, le numéro de la clef est invalide.
-8- fin de fichier.
-12- problème de verrouillage d'un enregistrement.
-16- toute autre raison (noyau FBSS non chargé...).

‹ENREG› contient l'enregistrement passé et renvoyé par ICEFIC.

La définition COBOL de ces zones d'interface est proposée en annexe sous l'intitulé "WCPRBBLK.CBL". [Annexe 2.1.1.]

La manière précise dont ICEFIC réalise son travail, c'est-à-dire les algorithmes qu'il utilise et les problèmes d'interfaçage avec FBSS, sont des détails d'implémentation. Ils sont exposés dans la partie consacrée à la phase de prototypage.

b. Conception du précompilateur ICE-PC

La réalisation du module ICEFIC permet d'écrire des applications en termes des primitives de ce module. Pour que ces applications ressemblent à celles qui sont écrites pour le mainframe, il doit exister un précompilateur qui soit capable de traduire les programmes ICE-PC en COBOL standard. Nous présentons ici les spécifications du précompilateur ICE-PC que nous avons développé à cette fin.

Rappelons que ce précompilateur ne traite que des programmes réalisant des transactions. Les concepts de transaction et de programme transactionnel ont été définis dans le troisième chapitre [Cfr III - 4].

Le précompilateur ICE-PC doit rendre principalement quatre services au développeur :

- il transforme la structure ICE en un programme COBOL.
- il assure l'intégrité des fichiers.
- il dispense de la déclaration des fichiers.
- il standardise l'interface hommme-machine.

Spécifions maintenant plus précisément chacun de ces services.

- La transformation de la structure

La description des programmes transactionnels en ICE est calquée sur celle du concept de programme transactionnel [Cfr III - 4].

Rappelons qu'un programme transactionnel ICE est décrit à l'aide d'une suite de paragraphes particuliers; il s'agit des paragraphes : ICEIPR, ICEITR, ICE001-I, ICE001-T, ICE001-E, ..., ICETTR, ICETPR.

Tout programme ICE-PC est structuré de la façon suivante :

```
ICEIPR (Initialisation du programme)


ICEITR (Initialisation de transaction)

   [
     ICE001-I (initialisation écran iii)

     ICE001-T (initialisation écran iii)

     ICE001-E (initialisation écran iii)
   ]

   [...]

   ICETTR (Terminaison de transaction)



ICETPR (Terminaison du programme)
```

Fig. IV - 8

Le rôle du précompilateur ICE-PC est, de transformer le texte de l'application ICE en un programme COBOL compilable. Le précompilateur doit donc insérer des instructions de branchement (GOTO) au bon endroit du programme.

Chacun connaît les dangers induits par l'utilisation de l'instruction "GOTO". Rappelons le texte de Dijkstra "GOTO STATEMENT CONSIDERED HARMFUL" [DS68]. Il est intéressant de noter que le programmeur d'applications ne doit jamais utiliser de "GOTO". Il ne doit ni s'occuper de réaliser une boucle sur la transaction, ni demander à l'utilisateur s'il désire mettre fin au programme. Ces différentes tâches sont assurées par les instructions rajoutées par le précompilateur ICE-PC.

- L'intégrité des fichiers

Si une première fonction du précompilateur est de transformer la structure des programmes ICE en une structure COBOL, une seconde consiste à assurer l'intégrité des fichiers.

Le programmeur n'a pas à se préoccuper des incohérences qui pourraient survenir suite à un évènement imprévu (par exemple, une coupure de courant ou l'abandon par l'utilisateur, soit d'une transaction soit du programme). C'est au précompilateur de rajouter les instructions qui garantissent l'intégrité des fichiers.

Il doit en particulier rajouter un "BEGIN TRANSACTION" après l'initialisation de chaque transaction (c'est-à-dire avant le paragraphe ICEITR). D'autre part il doit aussi réaliser un "END TRANSACTION" (l'équivalent du "COMMIT") après chaque transaction (c'est-à-dire après le paragraphe ICETTR).

La règle peut s'énoncer comme suit : toute transaction commence par un "BEGIN TRANSACTION" et n'est physiquement exécutée que lors du "END TRANSACTION". De cette façon, si un évènement quelconque vient interrompre une transaction non encore confirmée par le "END TRANSACTION", celle-ci est annulée et n'est pas prise en compte au niveau des fichiers. Un mécanisme de "ROLL BACK" assure, pour sa part, l'annulation de toutes les opérations depuis la dernière initialisation de transaction ("BEGIN TRANSACTION").

- La déclaration des fichiers

Une troisième fonction du précompilateur est de soulager le programmeur de la déclaration des fichiers. Le développeur d'application n'a plus besoin de déclarer les fichiers qu'il désire utiliser. C'est au précompilateur à insérer les instructions nécessaires pour que le texte COBOL généré soit compilable. En réalité, les fichiers ICE ne sont pas considérés comme des fichiers COBOL. Il n'est donc plus nécessaire de les déclarer comme tels. Tous les fichiers utilisables sont des fichiers FBSS auxquels le programmeur accède par l'intermédiaire des primitives du module ICEFIC. Ces fichiers FBSS doivent être déclarés une fois pour toutes à l'aide d'un logiciel interactif fourni dans l'environnement FBSS.

- La standardisation de l'interface homme-machine

Enfin, la dernière des fonctionnalités du précompilateur est qu'il se charge de rajouter les instructions nécessaires à la standardisation de l'interface homme-machine.

Notamment, trois "touches fonctions" possèdent systématiquement la même signification pour l'utilisateur final.

"PF2", lui permet à tout moment de signaler qu'il désire abandonner la transaction en cours et en recommencer une nouvelle. L'abandon d'une transaction provoque un "ROLL-BACK" et un branchement vers le début du paragraphe "ICEITR".

"PF3", lui permet à tout moment de signaler qu'il désire abandonner le programme en cours. Précisons qu'il s'agit du seul moyen mis à sa disposition pour quitter proprement un programme. L'abandon d'une transaction provoque un "END-TRANSACTION" et un branchement vers le début du paragraphe "ICETPR".

"PF1", lui permet de confirmer la validité d'un écran qui lui est présenté. Confronté à un écran nécessitant une confirmation, l'utilisateur ne peut faire que trois choses :

1) appuyer sur "PF2" ou "PF3" avec les conséquences habituelles que l'on sait.

2) appuyer sur "PF1", il confirme alors la validité de l'écran qui lui est proposé. Dans ce cas le programme continue à s'exécuter normalement.

3) appuyer sur une autre touche, il infirme alors la validité de l'écran qui lui est proposé. Dans ce cas le programme réaffiche l'écran en question et en permet la modification.

- L'algorithme global

En conclusion de ce qui précède, le précompilateur ICE-PC doit transformer le texte de l'application ICE en un programme COBOL compilable. Il doit insérer des instructions permettant d'assurer la réalisation des services suivants :

- il transforme la structure ICE en un programme COBOL,
- il assure l'intégrité des fichiers,
- il dispense de la déclaration des fichiers,
- il standardise l'interface hommme-machine.

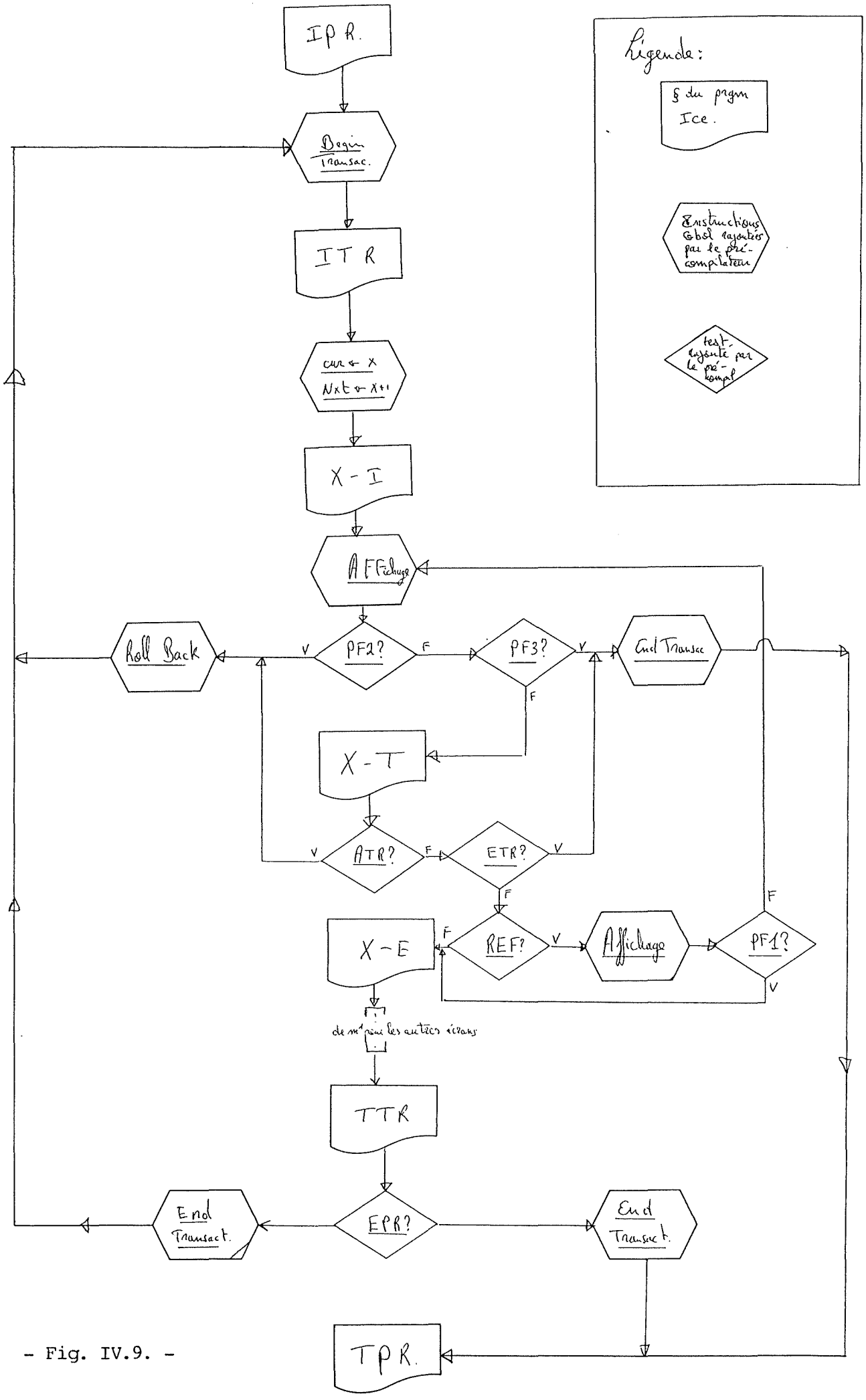
Le texte d'une application ICE doit être transformé selon le schéma de la figure IV - 9.

Il est important de comprendre que les instructions rajoutées entre les paragraphes X-I et X-E, c'est-à-dire les instructions de gestion, de gestion des écrans (affichage, saisie, validation, touches fonction, ...) doivent être rajoutées pour chaque écran. Il faut donc rajouter ces mêmes instructions entre les paragraphes Y-1 et Y-E, Z-I et Z-E, etc. si ces derniers existent.

3.3. Réalisation d'un prototype

Contrairement à la phase de conception globale, la phase de prototypage a été élaborée selon une méthodologie "bottom up". Nous avons réalisé une suite de sous-systèmes emboîtés. Cette approche nous a permis d'obtenir à la fin du travail un système peut-être limité mais qui fonctionne.

Nous proposons maintenant une description cohérente des principaux programmes réalisés. Les listings correspondants sont systématiquement reproduits et commentés en annexe.


IP R.
Begin Transac.
ITR
cur ← X
Nxt ← X+1
X - I
AFFichage
Roll Back
V
PF2?
F
PF3?
V
End Transac
F
X - T
V
ATR?
F
ETR?
V
F
X - E
F
REF?
V
Affichage
PF1?
F
V
de même pour les autres écrans
TTR
End Transact.
EPR?
End Transact.
TP R.
légende :
§ du prgm Ice.
Instructions Cobol rajoutées par le pré-compilateur
test représenté par le pré-compil

- Fig. IV.9. -

3.3.1. Interface COBOL-FBSS

Le premier développement que nous avons réalisé est un ensemble de procédures COBOL qui permettent d'utiliser les services de FBSS à partir d'une application écrite en COBOL.

L'interface entre FBSS et les applications est assurée par l'intermédiaire d'une zone de données communes aux deux logiciels. Cette zone appelée le "CPRBBLK".

```
01  CPRBBLK.

05  UERCPRB.

***  PARAMETRES D'ENTREE - INCHANGES PAR UN SEND-REQUEST   ***

   10  UERSERVER            USAGE IS POINTER.
   10  UERFUNCT             PIC X(2).
   10  UERQPARML            PIC 9(4) USAGE IS COMP-5 VALUE IS 26.
   10  UERQPARMAD           USAGE IS POINTER.
   10  UERQDATAL            PIC 9(4) USAGE IS COMP-5 VALUE IS 0.
   10  UERQDATAAD           USAGE IS POINTER.
   10  UERRPARML            PIC 9(4) USAGE IS COMP-5 VALUE IS 26.
   10  UERRPARMAD           USAGE IS POINTER.
   10  UERRDATAL            PIC 9(4) USAGE IS COMP-5 VALUE IS 0.
   10  UERRDATAAD           USAGE IS POINTER.

***  PARAMETRES DE SORTIE                                    ***

   10  UERRETCODE           PIC X(4).
   10  UERSERVRC            PIC X(4).
   10  UERREPLDPLEN         PIC 9(3) USAGE IS COMP-5 VALUE IS 0.
   10  UERREPLDDLEN         PIC 9(3) USAGE IS COMP-5 VALUE IS 0.
***  PCI-01 16-09-88 : Return PC ID to applications       ***
   10  UERPCID              PIC X(2).

***  PARAMETRES OPTIONNELS SUPPLEMENTAIRES                 ***

   10  UERUSERID            PIC X(10).
   10  UERPASSWORD          PIC X(10).
   10  UER3270IND           PIC 9.
```

```
***  RESERVE A FBSS                                          ***

   10  UERRSVD1            PIC X(129).

*
 05 REQPARMAREA              PIC X(26).
 05 REQ-PARM REDEFINES REQPARMAREA.
     10 OPERATOR              PIC X(2).
     10 PCBNAME               PIC X(8).
     10 FILLER                PIC X(16).
 05 REPPARMAREA              PIC 9(3).
 05 REQ-DATA                 PIC X(512).
 05 REP-DATA                 PIC X(512).
```

Fig. IV - 10

La spécification précise des zones est fournie dans [PRMS].

L'appel à une fonction de FBSS s'effectue en initialisant les paramètres nécessaires de la zone d'interface "CPRBBLK" (Cfr. exemple plus bas les instructions de 1 à 4) puis en exécutant un "CALL" à une routine de FBSS (instruction 5). Par exemple, le programmeur demande à FBSS de réaliser un "BEGIN TRANSACTION" de la manière suivante :

```
FBSS-BT.

MOVE "TB"        TO UERFUNCT.                    (1)
MOVE "         " TO PCBNAME OF REQ-PARM.         (2)
MOVE "  "        TO OPERATOR OF REQ-PARM.        (3)
MOVE 0           TO UERQDATAL UERRDATAL.         (4)
CALL "_SEND_REQUEST" USING UERCPRB.              (5)
```

Fig. IV - 11

(1) on désire utiliser la fonction FBSS : "TB" (Transaction Begin)
(2) il n'y a pas de PCBNAME
(3) il n'y a pas d' OPERATOR
(4) la longueur de la zone de données est 0
(5) appel à une routine FBSS

Nous avons créé un fichier qui contient une série de paragraphes COBOL. Chaque paragraphe porte un nom. Ce nom commence toujours par "FBSS-" et est suivi de 2 lettres qui identifient la fonction de FBSS que ce paragraphe réalise. Par exemple, "FBSS-BT" est le paragraphe COBOL qui réalise le "BEGIN TRANSACTION" de FBSS [Cfr supra].

Les fonctions de FBSS disponibles à partir d'une application COBOL sontles suivantes :

```
FBSS-GF. : GRANT FUNCTION ouvre le serveur de fichiers

FBSS-RF. : REVOKE FUNCTION ferme le serveur de fichier

FBSS-OS. : OPEN SESSION ouvre une session supplémentaire

FBSS-CS. : CLOSE SESSION ferme une session préalablement ouverte

FBSS-OB. : OPEN BATCH ouvre une session en mode batch

FBSS-CB. : CLOSE BATCH ferme une session ouverte en mode batch

FBSS-OO. : OPEN ONLINE ouvre une session en mode online

FBSS-BT. : BEGIN TRANSACTION marque le début d'une transaction

FBSS-CP. : CHECKPOINT valide toutes les opérations réalisées
           durant la transaction en cours

FBSS-ET. : END TRANSACTION marque la fin d'une transaction

FBSS-RB. : ROLL BACK annule toutes les opérations réalisées
           durant la transaction en cours

FBSS-GN. : GET NEXT lit l'enregistrement suivant du fichier
           spécifié

FBSS-GP. : GET PREVIOUS lit l'enregistrement précédent du fichier
           spécifié

FBSS-GU. : GET UNIQUE lit l'enregistrement correspondant à une clef
           dans un fichier spécifié
```

FBSS-RH. : READ HEADER fournit les informations sur un fichier de PCB donné

FBSS-DL. : DELETE RECORD détruit l'enregistrement d'un fichier donné

FBSS-HN. : HOLD NEXT bloque l'enregistrement suivant d'un fichier spécifié

FBSS-HP. : HOLD PREVIOUS bloque l'enregistrement précédent d'un fichier spécifié : FBSS-HP.

FBSS-HU. : HOLD UNIQUE bloque l'enregistrement spécifié par sa clef d'un fichier spécifié

FBSS-IS. : INSERT RECORD insère un enregistrement dans un fichier

FBSS-RP. : REPLACE RECORD modifie les valeurs d'attribut d'un record

FBSS-EX. : REQUEST EXCLUSIVE USE requiert l'usage exclusif d'un DBD

FBSS-HL. : HABILITATE LOG autorise

FBSS-IL. : INHIBIT LOG permet de charger des fichiers partagés

FBSS-IP. : RESET POINTERS initialise les pointeurs du serveur de fichiers

FBSS-SR. : STATISTIC REQUEST

FBSS-TS. : TEST STATUS teste si le serveur de fichiers est ouvert ou fermé

FBSS-INIT-CPRB-DB. : INIT initialise les pointeurs du serveur de fichiers

Fig. IV - 12

En annexe [Cfr Annexe - 2.1.2.] on trouvera, sous l'intitulé "FBSS-FCT.CBL", le fichier qui contient l'ensemble des paragraphes décrivant chacune de ces fonctions d'interface FBSS-COBOL.

3.3.2. FIC (ICEFIC-PC)

Le deuxième programme réalisé est le module d'accès aux données "ICEFIC" décrit précédemment. Dans l'environnement PC il porte le nom de "FIC".

Le programme "FIC" doit assurer l'exécution des fonctions dont la syntaxe a été décrite durant la conception détaillée du module ICEFIC [Cfr IV - 3.2.2. a].

L'algorithme du programme "FIC" peut être décrit brièvement de la manière suivante :

- il initialise le CPRB (zone d'interface avec FBSS),
- il réalise le lien entre noms logique et physique [*],
- il teste l'existence d'erreur syntaxique dans l'appel,
- il réalise l'accès demandé via FBSS,
- il analyse les codes d'erreur FBSS,
- il génère les codes d'erreur ICE-PC.

Fig. IV - 13

[*] Le programmeur fait référence à un nom logique de fichier. Le programme RPT se charge d'accéder à un fichier technique de l'environnement ICE-PC. Ce fichier technique est l'équivalent du répertoire des fichiers de l'environnement ICE sur le mainframe. Le nom logique des fichiers est la clef d'accès de ce répertoire. Il associe ce nom logique à l'ensemble des caractéristiques techniques du fichier (longueurs des enregistrements, longueurs des clefs, ...).

Les listings des programmes "FIC" et "RPT" sont reproduits en annexe sous les intitulés "RPT.CBL" et "FIC.CBL".
[Cfr Annexe - 2.2.]

3.3.3. Le précompilateur ICE-PC

Le dernier développement réalisé est le précompilateur ICE-PC. Il a été réalisé en TURBO-PASCAL.

Dans sa version actuelle, il réalise les spécifications décrites au point 3.2.2.b du chapitre IV : il insére des instructions permettant d'assurer la réalisation des services suivants :

- il transforme la structure ICE en un programme COBOL,
- il assure l'intégrité des fichiers,
- il dispense de la déclaration des fichiers,
- il standardise l'interface hommme-machine.

De plus il reconnaît quatre variables prédéfinies de l'environnement ICE, à savoir :

1) ICECUR (donne le numéro de l'écran courant),
2) ICENXT (donne le numéro du prochain écran),
3) ICEATR (provoque un ROLL BACK),
4) ICEETR (exige une confirmation de l'écran courant).

Le schéma standard des transformations que le précompilateur fait subir à un programme d'application écrit en "ICE" est le même que celui décrit à la figure IV - 9. Il faut cependant y rajouter le traitement des variables de l'environnement ICE (ICECUR, ICENXT, ICEATR, ICEETR).

Le précompilateur utilise quatre fichiers techniques :

1) le fichier des messages qui contient des parties de code COBOL qui sont ajoutées au programme ICE,

2) le fichier des erreurs qui contient des messages d'erreur générés par le précompilateur,

3) le fichier source qui contient le programme d'application ICE à précompiler,

4) le fichier output qui contient le programme COBOL résultant de la précompilation.

Le principe général de fonctionnement du précompilateur est le suivant. Il parcourt le fichier source qui contient le texte de l'application en ICE et le recopie en apportant certaines modifications dans le fichier output qui contient alors le texte COBOL de l'application correspondante.

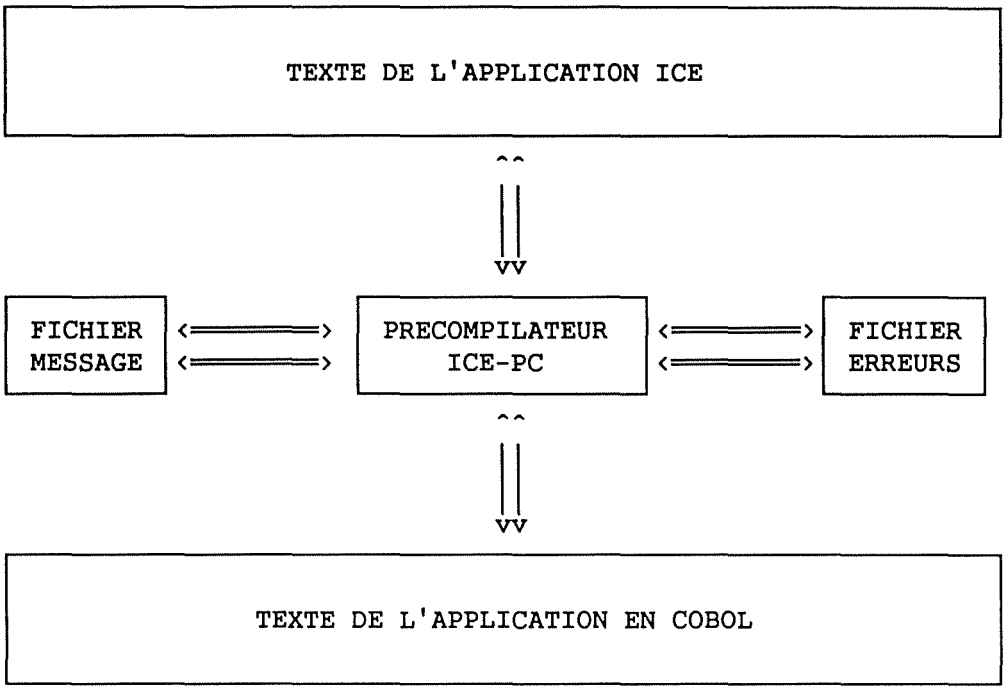


Fig. IV - 14

Les parties fixes du code COBOL à rajouter dans le fichier output sont puisées dans le fichier des messages. Il s'agit par exemple de la partie "ENVIRONMENT DIVISION" et "DATA DIVISION" qui contiennent toujours les mêmes déclarations de variables. Le précompilateur recopie donc au bon endroit les lignes de COBOL nécessaires.

Les parties variables du code COBOL, c'est-à-dire les parties qui ne sont pas identiques d'application à application, sont "calculées" par le précompilateur avant d'être insérées dans le fichier "output".

Lors de son parcours du fichier "source", le précompilateur peut découvrir une incohérence. Par exemple, il se peut que le programmeur ait oublié un paragraphe ICE. Dès que cette erreur est découverte par le précompilateur, il affiche le message correspondant à l'erreur avant d'abandonner la précompilation.

De manière générale, le précompilateur recherche des en-têtes de paragraphes COBOL et insère, avant ou après, les lignes nécessaires.

Par exemple, à chaque précompilation, des parties fixes de codes sont insérées après le label "ENVIRONMENT DIVISION". Les lignes rajoutées permettent d'indiquer au système que la variable ICECRT doit contenir successivement le code des touches enfoncées par l'utilisateur final. Ce mécanisme transparent pour le développeur d'application ICE, permet de tester si les touches "PF1", "PF2" ou "PF3" ont été utilisées et donc de déclencher les procédures correspondantes [Cfr IV - 3.2.2.b].

De même, le précompilateur rajoute après le label "DATA DIVISION" la définition des variables ICE accessibles au développeur d'application (ICECUR, ICENXT, ICEATR, ICEETR). Il faut aussi déclarer à cet endroit les variables techniques de ICE. Celles-ci ne sont pas accessibles au développeur d'application. Il s'agit par exemple de la variable ICECRT dont nous avons déjà parlé plus haut.

Toujours de la même façon, après le paragraphe descripteur de la terminaison du programme (ICETPR), le précompilateur vient rajouter deux groupes de lignes COBOL.

Le premier groupe forme le paragraphe ICEAFF qui :

1) gère l'affichage des écrans,
2) assure la gestion des touches fonctions,
3) rebranche vers le paragraphe qui doit suivre.

Le second groupe forme les paragraphes de définition des fonctions FBSS en COBOL. Ces fonctions sont utilisées par le module ICEFIC mais sont aussi des fonctions qui sont rajoutées dans l'application par le précompilateur (BEGIN TRANSACTION, END TRANSACTION, ROLL-BACK, ...). Ces fonctions sont celles de l'interface COBOL-FBSS décrites précédemment.
[Cfr IV - 3.1.1.]

Le listing du précompilateur et des fichiers techniques est reproduit en annexe sous les noms "ICE.PAS" "MESSAGE.ICE" et "ERREUR.ICE". [Cfr Annexe - 2.3.]

3.3.4. Une application

Rappelons que nous devions produire un environnement de développement sur PC. La conception et la réalisation d'applications dans cet environnement n'étaient pas le but recherché.

A titre d'exemple, nous avons développé un programme d'application ICE. Ce programme "AFIC" permet de réaliser de façon interactive un appel aux fonctions du module "ICEFIC".

"AFIC" est donc un programme transactionnel. Une transaction de ce programme permet à un utilisateur final d'exécuter de façon interactive une primitive du module ICEFIC.

A titre de comparaison des langages ICE et COBOL, nous avons développé la même application en COBOL.

Nous reproduisons en annexe [Cfr Annexe - 2.4.] les différents listings de ces programmes, à savoir :

- langage ICE (AFIC.ICE),
- "AFIC" en COBOL obtenu après précompilation de "AFIC.ICE" ("AFIC.CBL"),
- "AFIC" en COBOL ("AFIC_ORIG.CBL").

4. CONCLUSION DU PROJET ICE-PC

Toutes les évaluations concernant ce projet ont été regroupées au chapitre suivant, nous pouvons cependant déjà retenir quelques conclusions de la conception et de la réalisation du prototype de ce projet.

1. Le projet générique consiste à produire une version PC de l'environnement ICE.

2. Il existe différentes façons de réaliser ce projet générique.

3. Une première solution consiste à adapter DBASE aux besoins du projet. Cette solution présente l'avantage d'être assez simple à mettre en oeuvre. Par contre, on a constaté qu'il subsistait de nombreux "bugs" dans ce logiciel, ce qui le rend peu fiable. Ensuite, DBASE est assez "gourmand" en mémoire centrale ce qui augmente fortement le prix du poste de travail.

4. Une autre solution consiste à utiliser FBSS. Cette solution est plus complexe à mettre en oeuvre que la précédente. Mais elle a l'avantage d'être moins "gourmande" en mémoire centrale et de fournir un résultat plus proche de l'environnement mainframe.

   Mais à nouveau, comme dans la solution DBASE, la solution FBSS semble peu fiable et peu portable à long terme. Enfin, n'oublions pas que nous n'avons développé qu'un prototype et que la réalisation d'un produit définitif demanderait d'importants investissements humains.

Tentons dans le chapitre suivant, de synthétiser les différentes évaluatiques que l'on pourrait faire de ce projet.

CHAPITRE V : ETUDE D'OPPORTUNITE DU PROJET ICE-PC

CHAPITRE V : Etude d'opportunité du projet ICE-PC

Au chapitre précédent nous avons présenté le projet ICE-PC et le développement de son prototype. Dans ce chapitre V, nous réalisons l'étude d'opportunité de ce projet.

"Une étude d'opportunité a pour but d'élaborer un avant-projet de solution à partir d'un examen approfondi des besoins exprimés. Elle porte essentiellement sur un examen du 'pourquoi' - c'est-à-dire sur les causes profondes des transformations à apporter au système d'information existant - et, en réponse, sur la nature des transformations à apporter à celui-ci..." [BO89].

Nous nous inspirons de la méthodologie proposée par Bodart et Pigneur dans [BO89]. Cette méthode est caractérisée par l'enchaînement de différentes étapes spécifiques. Les étapes d'identification du projet, de définition du projet cadre, l'étude critique de l'existant et la finalisation du projet cadre ont déjà été abordées aux chapitres précédents. Nous nous intéressons donc tout particulièrement aux étapes d'évaluation des solutions et de contrôle de la solution recommandée.

> La première "permet d'évaluer des solutions qui répondent aux éléments du projet cadre retenus par les responsables de l'organisation" [BO89].
>
> La seconde, l'étape de contrôle de la solution recommandée, "doit permettre aux membres du comité directeur du projet de prendre une décision d'acceptation ou de formuler des directives pour la recherche d'une nouvelle solution" [BO89].

Concrètement, nous proposons de réaliser :

- une évaluation technique du projet,
- une évaluation du risque du projet,
- une évaluation organisationnelle du besoin de décentralisation,
- une évaluation économique,
- une conclusion des évaluations qui précèdent.

1. EVALUATION TECHNIQUE DU PROJET

Le but de cette section est d'évaluer la faisabilité technique du projet. Il faut déterminer si le cadre proposé au chapitre précédent est oui ou non réalisable. Les premières analyses réalisées avant et pendant l'implémentation du prototype nous poussent à répondre par l'affirmative. Cependant, il nous paraît important de formuler différentes limites :

1) Il est difficile d'assurer sur PC le niveau de fiabilité (PANNES-ERREURS) atteint sur mainframe. Les premières versions de FBSS [4] étaient limitées et entachées de quelques erreurs. Pour y remédier, IBM propose le remplacement de ces versions par de nouvelles. Malheureusement, les interfaces des versions successives de FBSS ne sont pas compatibles. De plus, IBM annonce les nouvelles versions de son logiciel pour les nouveaux systèmes d'exploitation (DOS 4 et OS/2 [5]). Sur base des expériences précédentes, il est fort à craindre qu'à nouveau, les interfaces de ces futures versions ne soient pas compatibles. Ce haut degré d'incertitude et de modification des couches logicielles inférieures rendent le système fragile et peu portable à long terme.

2) Dans le même ordre d'idées, il semble difficile d'assurer sur un PC le niveau de sécurité (MALVEILLANCE) atteint sur le mainframe. Par exemple, il est difficile de garantir qu'un utilisateur éclairé n'ira jamais "visiter" les fichiers FBSS. Même si la sécurité peut être assurée par des techniques de chiffrage, l'effort nécessaire pour la garantir est loin d'être négligeable. Cette contrainte de sécurité est pourtant capitale dans une institution bancaire. Ce manque de fiabilité et de sécurité semble inacceptable à la Belgolaise comme dans la plupart des organisations sérieuses.

3) D'autre part, du fait de sa conception calquée directement sur le modèle mainframe, l'environnement développé sur PC possède les mêmes défauts que son aïeul (ICE-MAINFRAME [3]). D'abord par une certaine lourdeur. Par exemple, l'interface homme-machine tire très peu parti des qualités intrinsèques des micros. Il n'existe ni souris, ni multi-fenêtrage. L'utilisateur final ne peut que très difficilement passer d'une application ICE-PC à une application plus classique (tableur ou traitement de texte). Cette possibilité serait pourtant intéressante dans une vision bureautique intégrée du phénomène informatique. Elle permettrait par exemple, via une application transactionnelle ICE, d'extraire des données qui seraient ensuite manipulées sur un tableur, un traitement de texte ou un logiciel intégré.

Aujourd'hui en organisation, les PC sont le plus souvent utilisés à diverses tâches. Le succès de logiciels multi-tâches comme WINDOW illustre que la possibilité d'interrompre momentanément une application au profit d'une autre est une caractéristique très prisée des micros. De même, le succès de ces interfaces graphiques montre que l'existence de puissants mécanismes de désignation (souris) et le raffinement de l'interface Homme/machine sont perçus par les utilisateurs de PC comme une caractéristique primordiale d'un environnement PC.

L'adjonction de telles possibilités à l'environnement ICE sur PC apporterait une réelle valeur ajoutée par rapport à la version mainframe. L'amélioration de l'interface Homme/machine induit une plus grande convivialité. Cette dernière est souvent considérée comme l'un des facteurs ayant conduit au succès des ordinateurs personnels.

Par contre, le fait de rajouter dans l'environnement PC des fonctionnalités inexistantes sur le mainframe a comme conséquence directe d'éloigner, l'une de l'autre, les deux versions. Ce dernier élément va à l'encontre de la philosophie de standardisation qui nous a guidé jusqu'ici.

Enfin, si renoncer à la souplesse des PC peut apparaître intéressant d'un point de vue sécurité, cela soulève le problème du coût du poste de travail. Pourquoi prendre un PC sans aucun de ses avantages si l'on peut faire la même chose à partir d'un terminal moins coûteux.

4) Le prototype ne recouvre qu'une petite partie des spécifications globales du projet. De plus, de nombreuses limitations ont été apportées. Sur les six interfaces du système mainframe, seule l'interface d'accès aux données et une version simplifiée du précompilateur ont été implémentées. L'effort pour réaliser une version complète de ICE-PC est loin d'être négligeable. La version mainframe de ICE a nécessité pour son développement environ 4 années/Homme. On peut donc raisonnablement évaluer celui de la version PC à au moins deux années/Homme. [Cfr V - 4]

   En résumé, étant donné la mise au point d'un prototype et les remarques que nous venons d'émettre, le projet de décentralisation nous paraît techniquement faisable mais nécessite un important investissement de développement. Insistons aussi sur le fait qu'il ne tire que peu profit des atouts spécifiques du PC. Et que le système semble fragile pour des raisons de fiabilité et de portabilité du logiciel de base FBSS. Ces arguments ne sont pas suffisants pour valider l'opportunité d'un tel projet.

   Après avoir montré que le projet est réalisable sous certaines conditions, nous allons maintenant évaluer le risque qu'il présente pour l'organisation.

2. ESTIMATION DU RISQUE

Pour déterminer le risque lié au projet, nous suivrons la méthodologie proposée par Mc Farlane [MFRL].

Ce dernier utilise trois critères fondamentaux : la taille du projet, sa structure et la maîtrise de la technologie requise pour son développement. En croisant ces trois critères, il détermine 8 types de projet. Chaque type de projet est caractérisé par un niveau de risque particulier allant de "très faible" à "très élevé".

	STRUCTURE DU PROJET : ELEVEE	STRUCTURE DU PROJET : FAIBLE	
FAIBLE NIVEAU DE MAITRISE DE LA TECHNOLOGIE DU PROJET PAR LE CENTRE INFORMATIQUE	TAILLE : IMPORTANTE RISQUE : FAIBLE	TAILLE : IMPORTANTE RISQUE : F -> MODERE	
	TAILLE : MODESTE RISQUE : TRES FAIBLE	TAILLE : MODESTE RISQUE : FAIBLE	
HAUT NIVEAU DE MAITRISE DE LA TECHNOLOGIE DU PROJET PAR LE CENTRE INFORMATIQUE	TAILLE : IMPORTANTE RISQUE : ELEVE	TAILLE : IMPORTANTE RISQUE : TRES ELEVE	
	TAILLE : MODESTE RISQUE : F -> MODERE	TAILLE : MODESTE RISQUE : ELEVE	

Nous allons dans cette section, déterminer à quel type de projet appartient le développement de ICE sur PC.

Ce projet consiste à adapter, à un nouveau matériel, un logiciel que le département informatique a lui-même développé. Il est capable d'énumérer et de décrire les résultats attendus. On peut donc affirmer que ce projet est fortement structuré.

Ce projet concerne au premier chef le département informatique. L'équilibre de ce département ni celui d'aucun autre ne sont directement liés à la réussite de ce projet. Le montant des investissements qu'il nécessite [Cfr V - 4] (environ 18 mois/Homme) est assez faible en comparaison du budget du département informatique ou du chiffre d'affaires de l'entreprise. On peut donc considérer que ce projet est de petite taille.

Enfin, le personnel ne maîtrise pas parfaitement la technologie nécessaire. En effet, les informaticiens de la Belgolaise sont bien plus habitués à travailler sur du gros matériel que sur des PC. Le développement sur PC et en particulier en réseaux locaux est très limité. Dès lors, l'expérience de l'équipe de développement est très faible en ce domaine.

Le projet est donc fortement structuré, de taille modeste mais la maîtrise de la technologie est faible. Selon Mc Farlane, il s'agit là d'un projet dont le risque est faible voire modéré.

Après avoir évalué la faisabilité et le risque nous pouvons maintenant aborder les considérations organisationnelles liées au lancement d'un tel projet.

3. EVALUATION ORGANISATIONNELLE

Nous avons vu au chapitre I, que seuls certains services de la Belgolaise réalisaient leurs activités informatiques de manière décentralisée. Citons par exemple les services du Change et des Opérations Diverses qui réalisent les saisies sur des machines Texas-Instrument décentralisées [Cfr II - 1.5.].

Cette particularité n'est pas inhérente à l'activité de ces services mais résulte d'un ensemble de considérations historiques. Ces applications pourraient facilement être développées sur le mainframe. Il est donc inutile de les réaliser sur PC.

Enfin, la structure de la Belgolaise, son caractère centralisé, et principalement le fait qu'elle ne possède aucune agence démontre encore l'inexistence de besoins de décentralisation du traitement des transactions. [Cfr I]

Cependant, la centralisation intégrale du système informatique induit aussi un important défaut : en cas d'indisponibilité du mainframe, c'est l'ensemble du système informatique qui est paralysé.

La disponibilité d'applications complètement décentalisées et indépendantes du système central présenteraient un réel atout pour l'ensemble de l'organisation. Elle induirait notamment deux effets particulièrement intéressants :

- Des services comme les Opérations Diverses et le Visa, en cas d'indisponibilité du mainframe, pourraient continuer à assurer leur mission qui est vitale pour les clients et donc pour la Banque.

- De plus, les opérations de ces deux services qui fonctionnent actuellement sur les TEXAS, sont à faible valeur ajoutée. Leur transfert sur le mainframe induirait une forte augmentation de la charge CPU. Cette augmentation ne pourrait être supportée par la configuration informatique actuelle qu'il faudrait dès lors améliorer. Nous avons vu [Cfr I - 1.1.] que cette dernière opération était délicate et coûteuse. Le coût marginal (1 million par an [BL88]) à consentir pour centraliser ces deux applications à faible valeur ajoutée est donc inacceptable pour la Direction. La solution qui consisterait à décharger ces applications sur un réseau de PC est donc nettement plus intéressante financièrement parlant.

  D'autre part, il existe des besoins en moyens informatiques décentralisés, mais ceux-ci sont principalement d'ordre bureautique. Secrétaires et cadres utilisent déjà à profusion les matériels et logiciels de ce type. Ces besoins sont largement satisfaits par les applications bureautiques traditionnelles (tableur et traitement de texte).

4. EVALUATION ECONOMIQUE

La dernière des évaluations à réaliser dans ce chapitre est l'évaluation économique globale du projet. Il nous faut confronter les coûts et les recettes que le développement d'ICE-PC induit pour la Banque.

Commençons par évaluer le coût du développement de ICE :

ANALYSE : 4 mois/homme (m/h)
+ 1 m/h pour coordination sécurité, exploitation, ...
1 analyste = 250.000 Frs/mois (prix revient patronal)
=> 5 * 250.000 = 1.250.000. Frs

PROGRAMMATION : 4 m/h
1 programmeur = 150.000 Frs/mois (prix revient patronal)
=> 4 * 150.000 = 600.000. Frs

TEST : 4 m/h
1 analyste-programmeur = 200.000 Frs/mois
=> 4 * 200.000 = 800.000. Frs

OPTIMISATION : 4 m/h
1 analyste-programmeur = 200.000 Frs/mois
=> 4 * 200.000 = 800.000. Frs

ANALYSE :	1.250.000. Frs
PROGRAMMATION :	600.000. Frs
TEST :	800.000. Frs
OPTIMISATION :	800.000. Frs
TOTAL :	3.450.000. Frs

Les calculs qui précèdent montrent donc que le prix de revient pour la Belgolaise du développement de l'environnement ICE sur PC s'élève aproximativement à 3,5 millions de francs.

Nous pouvons dégager quatre types de coûts liés au développement de ce projet :

1) Coût du matériel
   (unité + cartes réseau + câbles + placement = 200.000 Frs)
   Serveur = 200.000 Frs
   Poste Travail = 80.000 (CPU) + 40.000 (réseau) = 120.000 Frs
   => pour 5 unités : 680.000 Frs

2) Coût des logiciels
   (OS, COBOL, FBSS)
   => 100.000 Frs

3) Coût du développement des logiciels, développement de ICE-PC 3.450.000 Frs [Cfr supra] développement des applications = 12 mois/Homme à 200.000 Frs
   => 3.450.000 Frs + 2.400.000 Frs = 5.850.000 Frs

4) Coût de la maintenance des logiciels : maintenance = 30%
   [Cfr III - 6.5.]
   => 2.400.000 Frs * 30/70 = 1.028.000 Frs

   En résumé :

1) Coût du matériel	680.000 Frs
2) Coût des logiciels	100.000 Frs
3) Coût du développement des logiciels	5.850.000 Frs
4) Coût de la maintenance des logiciels	1.028.000 Frs
TOTAL (pour 5 PC)	7.658.000 Frs

Nous arrivons pour l'ensemble du projet à une estimation totale de plus de 7,6 millions de francs. En réalité, le développement de ICE-PC seul ne revient qu'à environ 3.500.000. Frs [Cfr supra]. Il faut, pour bien faire, comparer cette somme aux recettes que l'on est en droit d'attendre de ce projet.

Les recettes peuvent être de deux types :

Le premier type de recettes est lié au développement d'une application sur PC. Il est possible qu'une application développée sur un PC soit à elle seule une recette. Par exemple, si le mainframe est saturé, la disponibilité d'une application sur un PC peut éviter l'achat ou le remplacement d'un mainframe. La décentralisation en elle-même peut donc induire des recettes par diminution des coûts.

Le second type de recettes est lié au développement de ICE-PC. Ce dernier est, nous le savons, un environnement de programmation. Le but d'un tel environnement est d'aider le développeur dans sa tâche. ICE-PC induit donc un accroissement de productivité des acteurs du développement. Mais comment évaluer l'augmentation de productivité d'un développeur ?

Pour mesurer scientifiquement cet accroissement de productivité, il faudrait pouvoir comparer diverses choses. Par exemple, il faudrait pouvoir comparer le temps de développement d'une même application par un même programmeur, qui travaillerait d'abord sans ICE puis avec ICE. Mais cette mesure serait faussée car à la seconde fois, l'informaticien bénéficierait immanquablement de l'expérience de la première implémentation. De même la mesure serait faussée si on utilisait deux personnes différentes. Enfin, il est difficile de comparer la complexité de la programmation en fonction des langages utilisés, car des critères tels que le nombre d'instructions sont assez peu fiables et traduisent mal l'effort de création.

Face à ces difficultés méthodologiques, nous proposons de recourir à une hypohèse simplificatrice. Faisons l'hypothèse que l'environnement ICE-PC double la productivité des programmeurs. Le développement des applications coûtera donc deux fois moins cher.

Cette hypothèse est plus ou moins vérifiée puisque la part de la maintenance dans le cycle de vie d'un projet passe de 60 % [BM84] à 30 % [BL88].

Les recettes induites par le développement de ICE-PC seront donc fonction du nombre d'applications développées dans cet environnement. Plus la Belgolaise développe d'applications avec ICE-PC, plus elle rentabilise le coût de développement de cet atelier logiciel.

Essayons de déterminer à partir de quelle quantité de logiciels, la Belgolaise a intérêt à développer ICE.

Dans un premier temps, nous allons considérer que les recettes du premier type (diminution des coûts par décentralisation) sont nulles. Développer ICE-PC n'est alors intéressant que si le coût de développement des applications sans ICE-PC est supérieur à celui des applications à l'aide de ICE-PC plus celui de ICE-PC.

Développer ICE-PC est intéressant si et seulement si :

$$SSI > SAI$$

SSI = solution sans ICE
SAI = Solution avec ICE

$$SAI = SSI$$
$$\Rightarrow DI + DAI = DSI$$
$$\Rightarrow DI + \frac{DSI}{2} = DSI$$
$$DI = DSI - \frac{DSI}{2}$$
$$= DSI \left(1 - \frac{1}{2}\right)$$
$$= \frac{DSI}{2}$$

DI = coût du développement de ICE
DSI = coût du développement des applications sans ICE
DAI = coût du développement des applications avec ICE = $\frac{DSI}{2}$

Si la Belgolaise développe pour plus de 7 millions d'applications, le développement de ICE devient économiquement intéressant. En effet, le développement des applications sans ICE lui coûterait par hypothèse 7 millions, tandis que le développement de ICE reviendrait à 3,5 millions auxquels il faudrait ajouter le développement des applications à l'aide de ICE. Nous avons fait précédemment l'hypothèse que ICE-PC doublait la productivité des développeurs. Le développement des applications à l'aide de ICE reviendrait donc à 3,5 millions. Le développement de ICE et celui des applications coûteraient alors la même chose que le développement des applications sans ICE (soit 7 millions).

En deçà de ce niveau critique de 7 millions de Frs, le développement de ICE n'est pas rentable. Ce niveau représente le point neutre de l'opération.

En faisant l'hypothèse que ICE multiplie par K la productivité des développeurs, on pourrait généraliser ce qui précède par la formule suivante :

Point_Neutre ssi : $DI = DSI \left(\frac{K-1}{K}\right)$

De façon générale, on obtient un point neutre, c'est-à-dire que l'on réalise une opération blanche, si le développement de ICE égale $\frac{K-1}{K}$ fois le développement des applications sans ICE.

Nous pouvons aussi lever la première hypothèse selon laquelle la décentralisation n'induit aucune diminution de coûts. On atteint alors un point neutre si et seulement si :

$$SM = SPC$$

SM = solution mainframe
SPC = solution PC

or SPC

$$= DSI - RD$$
$$= DI + DAI - RD$$

RD = recettes de décentralisation

Donc point neutre si :

$$SM = DI + DAI - RD$$

On peut admettre que développer une application sur le mainframe revient au même prix que de la développer sur PC dans la mesure où l'on travaille avec le même environnement ICE.

donc point neutre ssi :

$$DA = DI + DA - RD$$
$$\Rightarrow DI - RD = 0$$
$$\Rightarrow DI = RD$$

DA : dévellopement des applications

Nous arrivons donc à la conclusion que le développement d'applications décentralisées devient rentable si et seulement si les recettes induites par cette décentralisation sont supérieures aux coûts du développement de ICE-PC.

DI < RD

Or nous avons montré que les besoins d'exploitation décentralisée étaient presque inexistants. Toutes les applications transactionnelles dont la Belgolaise a besoin peuvent parfaitement être développées sur le mainframe. Le projet n'est donc rentable que si le coût du développement de ICE-PC est inférieur aux recettes de décentralisation.

Enfin, des recettes substantielles (3-4 millions [BL88]) pourraient être induites par la revente de ICE-PC et dès lors peser en faveur de sa réalisation complète.

De plus l'image de marque de la Belgolaise s'en verrait améliorée. Elle serait alors considérée par le marché comme un innovateur maîtrisant parfaitement les technologies de l'information.

Cependant, le niveau de raffinement que devrait atteindre ICE-PC pour pouvoir être revendu est probablement difficile à atteindre pour une équipe restreinte de développeurs. Enfin, la revente de logiciels n'entre pas dans les activités normales de la Banque.

5. CONCLUSIONS DE L'ETUDE D'OPPORTUNITE

En conclusion de ce chapitre nous pouvons affirmer :

1) Le projet est techniquement réalisable mais les investissements humains et techniques pour obtenir un niveau de fiabilité acceptable sont non négligeables.

2) Le risque inhérent à ce projet est modéré.

3) Il n'existe pas de besoins transactionnels qui ne puissent être satisfaits par le système centralisé actuel. Le projet de décentralisation des applications transactionnelles s'avère donc inutile.

   Cependant, la disponibilité d'applications décentralisées et indépendantes induirait une augmentation de la disponibilité et une diminution du coût marginal du système informatique global.

4) La rentabilité de ce projet dans le cadre limité de la Belgolaise semble actuellement faible.

En conséquence, même si le projet est techniquement réalisable, les considérations d'ordre économique et organisationnel, en d'autres termes, l'étude d'opportunité penche en faveur de l'abandon de ce projet dans le stricte cadre de la Belgolaise.

CHAPITRE VI : CONCLUSION GENERALE

CHAPITRE VI : Conclusion générale

Nous voici arrivé au terme de ce mémoire. Il est temps de proposer une synthèse des différents aspects que nous avons abordés jusqu'à maintenant :

La Belgolaise est une Banque de taille moyenne dont l'organisation est classique et centralisée. Sa stratégie consiste à conserver le leadership du marché africain où elle finance les transactions commerciales d'import/export, par l'intermédiaire de crédits documentaires. Pour maintenir sa position sur ce marché, elle se doit de rester compétitive face à ses concurrents.

Dans cette optique, l'informatique est une ressource stratégique car elle permet d'améliorer la compétitivité en diminuant les coûts de traitement des informations grâce à une automatisation des tâches opérationnelles.

La stratégie informatique est de développer des applications pour améliorer constamment le service rendu à la Banque et aux clients. Mais on constate parallèlement une volonté de contrôler les coûts de production de ces applications en augmentant la productivité des informaticiens.

Cet accroissement de productivité a été obtenu par la mise au point et l'utilisation systématique de méthodes et d'outils d'aide au développement qui ont été regroupé au sein d'un environnement de développement baptisé "ICE".

L'environnement "ICE", présente comme principal avantage de faciliter la factorisation du travail en encourageant la réutilisation d'outils prédéfinis. De plus, en assurant de nombreuses fonctions de façon implicite, ICE augmente la fiabilité des applications, la standardisation du développement et de l'interface homme-machine. Ces deux avantages cumulés induisent une diminution des coûts de développement et de maintenance qui engendre une augmentation de la productivité des informaticiens. Par contre, et c'est là sa principale faiblesse, ICE souffre d'un manque évident de support à l'analyse.

CHAPITRE VI : Conclusion générale

Le système informatique qui supporte l'environnement ICE est, à l'image de la Belgolaise, classique, centralisé et de taille moyenne. Ses gestionnaires sont actuellement confrontés à différents problèmes.

Le premier est celui de la dépendance excessive de l'organisation vis-à-vis de cet unique mainframe.

Le deuxième est celui de l'évolution de la configuration actuelle qui est en passe d'être saturée par la masse sans cesse croissante des opérations journalières à réaliser.

Le troisième problème est celui de l'augmentation des coûts logiciel et matériel induit par la croissance de la configuration informatique centrale.

En réponse à ces problèmes, la Belgolaise envisage de développer et d'exploiter sur un réseau de PC des applications qui étaient préalablement supportées par le mainframe.

Il serait alors possible de créer des applications sur PC avec les mêmes méthodes et dans le même formalisme que celui utilisé sur le mainframe. De plus, il serait aussi possible d'exploiter ces applications sur le réseau de PC.

La disponibilité d'applications complètement décentralisées et indépendantes du système central présenteraient un réel atout pour l'ensemble de l'organisation. Elle induirait notamment deux effets particulièrement intéressants :

- Des services comme les Opérations Diverses et le Change, en cas d'indisponibilité du mainframe, pourraient continuer à assurer leur mission qui est vitale pour les clients et donc pour la Banque.

- De plus, les opérations de ces deux services qui fonctionnent actuellement sur les TEXAS, sont à faible valeur ajoutée. Leur transfert sur le mainframe induirait une forte augmentation de la charge CPU. Cette augmentation ne pourrait être supportée par la configuration informatique actuelle qu'il faudrait dès lors améliorer. Cette dernière opération est délicate et coûteuse. Le coût marginal à consentir pour centraliser ces deux applications à faible valeur ajoutée est donc inacceptable pour la Direction. La solution qui consisterait à décharger ces applications sur un réseau de PC pourrait donc être plus intéressante financièrement parlant.

Mais, pour assurer un développement harmonieux et, à terme, une diminution des coûts de réalisation et de maintenance de ces applications sur les PC, la Belgolaise souhaite adapter ICE à ce réseau.

Cette adaptation de "ICE" est donc un préalable au projet de "DOWN SIZING". Nous avons dès lors réalisé une étude d'opportunité de ce projet. Ensuite nous avons mis au point un prototype de l'environnement baptisé "ICE-PC". Ce dernier utilise les services d'un nouveau produit IBM, "FBSS" qui est une couche logicielle d'extension du DOS dont la fonction principale est de gérer l'accès aux ressources partagées du réseau (fichiers, imprimantes, ...).

Les conclusions qui se dégagent de l'étude d'opportunité et de l'expérience de prototypage, sont que ICE-PC connaît des problèmes de fiabilité et de portabilité à long terme. Ces problèmes sont principalement dus au logiciel FBSS que nous avons utilisé comme base à ce projet. Parallèlement, la rentabilité de ce projet dans le cadre limité de la Belgolaise est encore actuellement faible. Pour ces raisons, il a été décidé que le projet serait abandonné.

Néanmoins, cette situation est susceptible d'évoluer. En effet, la diminution constante du coût des PC (PRIX/PUISSANCE) et l'émergence de logiciels réputés plus fiables et plus puissant que FBSS, pousse la Banque à repenser ce projet à partir de PC fonctionnant sous UNIX.

ANNEXES

1 Les services

1.1 Le Visa

Le service Visa est composé de trois sections:

Accueil clientèle:

Toutes opérations aux guichets (ouverture de comptes et matières connexes).

Carte de crédit:

Emission et modification (Eurocard, Amexco, Eurochèques).

Vérification des ordres:

Contrôle de la signature et de la provision préalablement à toute opération.

1.2 Les Opérations Diverses

Ce service effectue toutes les opérations en francs belges à l'étranger et en Belgique, telles que virements en compensation, virements à l'étranger via correspondant, domiciliation, ordres permanents, chèques circulaires, mandats et virements postaux, mandats à l'étranger en FB et en devises étrangères, de même que les encaissements de chèques et ordres de paiement vers l'étranger.

1.3 Les Titres

La mission du service Titres s'articule autour des points fondamentaux suivants:

Négocier

C'est à dire, réaliser l'achat, la vente et la souscription des titres.

Conserver

C'est-à-dire garder les Titres pour compte de la clientèle.

Payer

C'est-à-dire verser les intérêts et dividendes provenant desdits titres.

Régulariser

C'est-à-dire se charger par exemple, d'échanges, d'offres publiques d'achat, de recouponnement, etc...).

1.4 Le Change

Le service Change est chargé de l'exécution de toutes les opérations en monnaies étrangères, telles que transferts, dépôts, présentations de chèques à l'encaissement, travellers chèques, réglementation des changes.

1.5 Le Crédit Documentaire

La Banque offre aux importateurs et exportateurs un éventail de services qui peuvent faciliter leurs transactions internationales.

Parmi cet éventail de services, le crédit documentaire joue un rôle très précieux dans le commerce international et offre aux partenaires commerciaux la sécurité la plus forte.

En résumé, le crédit documentaire est une opération de banque ayant pour objet de faciliter le réglement des marchandises et consistant en un contrat par lequel une banque s'engage, d'ordre et pour compte d'un acheteur (donneur d'ordre), à règler au vendeur (bénéficiaire), le prix d'une marchandise contre remise de documents déterminés.

1.6 Le Secrétariat des Crédits

Ce service est chargé de l'étude et de la mise en oeuvre de l'ensemble de crédits à la clientèle et de certains crédits aux banques. Il étudie les demandes, établit les dossiers pour les

instances de décisions, notifie les crédits aux clients. Il est chargé de la surveillance des dossiers et garanties après mise à disposition des crédits en vue des échéances ou du renouvellement éventuel.

Il a également dans ses attributions la récupération des créances litigieuses et la communication des renseignements commerciaux à l'usage de la clientèle ou des correspondants.

1.7 La Comptabiblité Centrale

Ce service comprend trois différentes sections:

La section Comptabilité Générale:

centralise les informations financières de l'entreprise et génère les rapports de gestion pour le Comité de Direction et pour les autorités extérieures.

La section Comptabilité Tenue des Comptes:

- pour les comptes détenus par les particuliers, banques et sociétés auprès de la Belgolaise:

- calcul des intérêts

- renseignements au client sur la position de ses comptes

- envoi, duplicata des extraits de compte

- pour les comptes détenus par la Banque chez d'autres banquiers: réconciliation des écritures comptabiblisées par divers services (Change, Portefeuille, etc...) avec celles comptabilisées chez nos banquiers correspondants.

La section Assistance Technique:

Centralise les informations comptables des affiliations africaines (B.K., B.C.B.).

2 Les développements

2.1 L'interface FBSS-COBOL

2.1.1 Zone d'interface COBOL-FBSS (WCPRBBLK.CBL)

```
      ****************************************************************
      * Fichier : WCPRBBLK.CBL
      *
      * Contenu : Ce programme contient la Working Storage pour l'interface
      *           FBSS
      *
      ****************************************************************


       01  CPRBBLK.

         05  UERCPRB.

      ***  SUPPLIED PARAMETERS - NOT CHANGED BY SEND-REQUEST   ***

           10  UERSERVER           USAGE IS POINTER.
           10  UERFUNCT            PIC X(2).
           10  UERQPARML           PIC 9(4) USAGE IS COMP-5 VALUE IS 26.
           10  UERQPARMAD          USAGE IS POINTER.
           10  UERQDATAL           PIC 9(4) USAGE IS COMP-5 VALUE IS 0.
           10  UERQDATAAD          USAGE IS POINTER.
           10  UERRPARML           PIC 9(4) USAGE IS COMP-5 VALUE IS 26.
           10  UERRPARMAD          USAGE IS POINTER.
           10  UERRDATAL           PIC 9(4) USAGE IS COMP-5 VALUE IS 0.
           10  UERRDATAAD          USAGE IS POINTER.

      ***  RETURNED PARAMETERS                                 ***

           10  UERRETCODE          PIC X(4).
           10  UERSERVRC           PIC X(4).
           10  UERREPLDPLEN        PIC 9(3) USAGE IS COMP-5 VALUE IS 0.
           10  UERREPLDDLEN        PIC 9(3) USAGE IS COMP-5 VALUE IS 0.

      ***  PCI-01 16-09-88: Return PC ID to applications       ***
           10  UERPCID             PIC X(2).

      ***  OPTIONAL EXTENDED FUNCTION INFORMATION              ***

           10  UERUSERID           PIC X(10).
           10  UERPASSWORD         PIC X(10).
           10  UER3270IND          PIC 9.

      ***  RESERVED FIELD                                      ***

           10  UERRSVD1            PIC X(129).
```

```
*
   05 REQPARMAREA              PIC X(26).
   05 REQ-PARM REDEFINES REQPARMAREA.
       10 OPERATOR              PIC X(2).
       10 PCBNAME               PIC X(8).
       10 FILLER                PIC X(16).
   05 REPPARMAREA              PIC 9(3).
   05 REQ-DATA                 PIC X(512).
   05 REP-DATA                 PIC X(512).
```

2.1.2 Listing des fonctions d'interface COBOL-FBSS

```
*****************************************************************
* Fichier : FBSS-FCT.CBL
*
* Contenu : Ce programme contient les fonctions de FBSS
*
* Auteur : D. Bauchau.
*
* Date : 28 septembre 89.
*
*****************************************************************


*.............................................................*
* GRANT FUNCTION ouvre le serveur de fichiers.
*.............................................................*

 FBSS-GF.
    MOVE "FG"       TO UERFUNCT.
    MOVE 1          TO UERQDATAL.
    MOVE 'O'        TO REQ-DATA.
    MOVE 0          TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.


*.............................................................*
*  REVOKE FUNCTION ferme le serveur de fichier.
*.............................................................*

 FBSS-RF.
    MOVE "FR"       TO UERFUNCT.
    MOVE 0          TO UERQDATAL.
    MOVE 0          TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.............................................................*
* OPEN SESSION ouvre une session supplémentaire
*.............................................................*

 FBSS-OS.
```

```
    MOVE "SO"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* CLOSE SESSION ferme une session préalablement ouverte
*.....................................................................*

 FBSS-CS.
    MOVE "SC"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* OPEN BATCH ouvre une session en mode batch
*.....................................................................*

 FBSS-OB.
    MOVE "BO"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* CLOSE BATCH ferme une session ouverte en mode batch
*.....................................................................*

 FBSS-CB.
    MOVE "BC"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* OPEN ONLINE ouvre une session en mode online
*.....................................................................*

 FBSS-OO.
    MOVE "OO"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL UERRDATAL.
    MOVE ' '         TO REQ-DATA.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* BEGIN TRANSACTION marque le debut d'une transaction
*.....................................................................*

 FBSS-BT.
    MOVE "TB"        TO UERFUNCT.
    MOVE "         " TO PCBNAME OF REQ-PARM.
    MOVE "  "        TO OPERATOR OF REQ-PARM.
    MOVE 0           TO UERQDATAL UERRDATAL.
```

```
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* CHECKPOINT valide toutes les opérations réalisées durant
*           la transaction en cours
*.....................................................................*

 FBSS-CP.
    MOVE "PC"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* END TRANSACTION marque la fin d'une transaction
*.....................................................................*

 FBSS-ET.
    MOVE "TE"        TO UERFUNCT.
    MOVE "         " TO PCBNAME OF REQ-PARM.
    MOVE "  "        TO OPERATOR OF REQ-PARM.
    MOVE 0           TO UERQDATAL UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* ROLL BACK annule toutes les opérations réalisées durant
*           la transaction en cours
*.....................................................................*

 FBSS-RB.
    MOVE "BR"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    MOVE "         " TO PCBNAME OF REQ-PARM.
    MOVE "  "        TO OPERATOR OF REQ-PARM.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* GET NEXT lit l'enrengistrement suivant du fichier spécifié
*.....................................................................*

 FBSS-GN.
    MOVE "NG"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* GET PREVIOUS lit l'enrengistrement précédent du fichier spécifié
*.....................................................................*

 FBSS-GP.
    MOVE "PG"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.
```

```
*.....................................................................*
* GET UNIQUE lit l'enrengistrement correspondant à une clef dans un
*            fichier spécifié
*.....................................................................*

 FBSS-GU.
    MOVE "UG"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.


*.....................................................................*
* READ HEADER fournit les information sur un fichier de PCB donné
*.....................................................................*

 FBSS-RH.
    MOVE "HR"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 8           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* DELETE RECORD détruit l'enregistrement d'un fichier donné
*.....................................................................*

 FBSS-DL.
    MOVE "LD"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* HOLD NEXT bloque l'enregistrement suivant d'un fichier spécifier
*.....................................................................*

 FBSS-HN.
    MOVE "NH"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* HOLD PREVIOUS bloque l'enregistrement précédent d'un fichier spécifier
*.....................................................................*

 FBSS-HP.
    MOVE "PH"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* HOLD UNIQUE bloque l'enregistrement spécifier par sa clef
*              d'un fichier spécifier
*.....................................................................*

 FBSS-HU.
    MOVE "UH"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.
```

```
*.....................................................................*
* INSERT RECORD insere un enregistrement dans un fichier
*.....................................................................*

 FBSS-IS.
    MOVE "SI"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* REPLACE RECORD modifie les valeur d'attribut d'un record
*.....................................................................*

 FBSS-RP.
    MOVE "PR"        TO UERFUNCT.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* REQUEST EXCLUSIVE USE requière l'usage exclusif d'un DBD
*.....................................................................*

 FBSS-EX.
    MOVE "XE"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* HABILITATE LOG autorise
*.....................................................................*

 FBSS-HL.
    MOVE "LH"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* INHIBIT LOG permet de charger des fichiers partagés
*.....................................................................*

 FBSS-IL.
    MOVE "LI"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
    MOVE 0           TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* RESET POINTERS initialise les pointeurs du serveur de fichiers
*.....................................................................*

 FBSS-IP.
    MOVE "PI"        TO UERFUNCT.
    MOVE 0           TO UERQDATAL.
```

```
    MOVE 0          TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* STATISTIC REQUEST
*.....................................................................*

 FBSS-SR.
    MOVE "RS"       TO UERFUNCT.
    MOVE 0          TO UERQDATAL.
    MOVE 18         TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* TEST STATUS teste si le serveur de fichier est ouvert ou fermé
*.....................................................................*

 FBSS-TS.
    MOVE "ST"       TO UERFUNCT.
    MOVE 0          TO UERQDATAL.
    MOVE 18         TO UERRDATAL.
    CALL "_SEND_REQUEST" USING UERCPRB.

*.....................................................................*
* initialise les pointeurs du serveur de fichiers
*.....................................................................*

 FBSS-INIT-CPRB-DB.
    SET UERSERVER  TO ADDRESS OF DBSERVEUR.
    SET UERQPARMAD TO ADDRESS OF REQPARMAREA.
    SET UERQDATAAD TO ADDRESS OF REQ-DATA.
    SET UERRPARMAD TO ADDRESS OF REPPARMAREA.
    SET UERRDATAAD TO ADDRESS OF REP-DATA.
```

2.2 L'interface ICEFIC d'accès aux données

2.2.1 Listing du programme FIC en COBOL

```
*****************************************************************
* Fichier : FIC.CBL
*
* Contenu : Ce programme permet de réaliser des accès standart aux
*
*           fichiers FBSS.
*
* Auteur : D. Bauchau.
*
* Date : 25 octobre 89.
*
*****************************************************************


*---------------------------------------------------------------*
 IDENTIFICATION DIVISION.
*---------------------------------------------------------------*
   PROGRAM-ID. FIC.
   AUTHOR. D-BAUCHAU.



*---------------------------------------------------------------*
 DATA DIVISION.
*---------------------------------------------------------------*

*.............................................................*
 WORKING-STORAGE SECTION.
*.............................................................*


 COPY WDB-INIT.CBL.
 COPY WSVPSAIS.CBL.
 COPY WRPTSTRC.CBL.
 COPY LEXTSTRC.CBL.

 01 ERREUR PIC 9(1) VALUE 1.

 01 VARRET PIC X(2).


 01 VARNUM PIC 9(1).

 01 NOM-PCB.
     02 DEBPCB PIC X(1) VALUE "P".
     02 NUMPCB PIC X(1) VALUE "1".
     02 NOMPCB PIC X(6).

 01 VAR-ELEM.
     02 VARLON PIC 9(4).
```

```
     02 VARPOS PIC 9(4).

*.....................................................................*
 LINKAGE SECTION.
*.....................................................................*

 COPY WFICINTR.CBL.

*---------------------------------------------------------------------*
 PROCEDURE DIVISION USING FUNCTION, PARAM, DEB, FICH.
*---------------------------------------------------------------------*

 DEBUT-FIC.


*    initialisation des codes d'erreurs.
     MOVE "  " TO DEB.

*    initialisation du CPRB.
     PERFORM FBSS-INIT-CPRB-DB.

*    Vérification des erreurs de syntaxe dans l'appel
     IF NOT FB-READFIRST AND NOT FB-READSTEP AND NOT FB-READIND
            AND NOT FB-REWRITE AND NOT FB-DELETE AND NOT FB-WRITE
            AND NOT FB-BEGTRANS AND NOT FB-ENDTRANS
            THEN MOVE "0?" TO DEB
                 PERFORM ERROR01-SECTION.

* Premier branchement sur la fonction désirée.
     IF FB-BEGTRANS THEN PERFORM FCT-BEGTRANS.
     IF FB-ENDTRANS THEN PERFORM FCT-ENDTRANS.
     IF NOT FB-BEGTRANS AND NOT FB-ENDTRANS
         THEN PERFORM AUTRE-FCT.

*    Repérage et analyse des code d'erreur s'il y en a eut
     PERFORM ANALYSE-RET-CODE-SECTION.

*    Restitution de la reponse
     IF RETCOD = 0 THEN MOVE REP-DATA TO ENREG.

*    fin de la partie principale du programme.
     EXIT PROGRAM.


 AUTRE-FCT.
*    le parametre par défaut est "NL"
     IF ( NOT FB-LOCK ) THEN MOVE "NL" TO PARAM.

*    Appel au répertoire des fichiers.
     MOVE NOMFIC TO NOMLOGFI.
     MOVE 0 TO RETCOD.
     CALL "RPT" USING ENREGRPT, ERREUR.

*    Vérification que l'appel s'est bien passé
```

```
    IF ERREUR = 1 THEN MOVE "F?" TO DEB
                       PERFORM ERROR01-SECTION.

*   si la lecture du répertoire s'est correctement effectuée alors
    IF NUMCLE = SPACE MOVE "1" TO NUMCLE.

*   vérifier que indice est bien dans ses bornes.
    MOVE NUMCLE TO NUMPCB.
    IF NUMPCB > NOMCLE  THEN MOVE "K?" TO DEB
                             PERFORM ERROR01-SECTION.

*   Mise à jour du CPRB grâce aux renseignements du répertoire.
    MOVE NOMLOGFI TO NOMPCB.
    MOVE NUMCLE TO NUMPCB.
    MOVE NOM-PCB TO PCBNAME.
    MOVE NOM-SRVR TO DBSERVEUR.
    MOVE NUMCLE TO VARNUM.
    SET  INDICE TO VARNUM.
    MOVE ELEMENT (INDICE) TO VAR-ELEM.
    MOVE VARLON TO UERQDATAL.
    MOVE RECLON TO UERRDATAL.

*   mise en place de la clef.
    MOVE ENREG TO DONNEE.
    MOVE VARPOS TO POSSTR.
    MOVE VARLON TO LONSTR.
    CALL "EXT" USING LEXTSTRC.
    MOVE RESULT TO REQ-DATA.


*   Branchement sur la fonction désirée.
    IF FB-READFIRST THEN PERFORM FCT-READFIRST.
    IF FB-READSTEP  THEN PERFORM FCT-READSTEP.
    IF FB-READIND   THEN PERFORM FCT-READIND.
    IF FB-REWRITE   THEN PERFORM FCT-REWRITE.
    IF FB-DELETE    THEN PERFORM FCT-DELETE.
    IF FB-WRITE     THEN PERFORM FCT-WRITE.



ERROR01-SECTION.
    MOVE 16 TO RETCOD.
    EXIT PROGRAM.

FCT-READFIRST.
    IF FB-LOCK   THEN PERFORM FCT-READFIRST-LOCK.
    IF FB-NOLOCK THEN PERFORM FCT-READFIRST-NOLOCK.

FCT-READSTEP.
    IF FB-LOCK   THEN PERFORM FCT-READSTEP-LOCK.
    IF FB-NOLOCK THEN PERFORM FCT-READSTEP-NOLOCK.

FCT-READIND.
    IF FB-LOCK   THEN PERFORM FCT-READIND-LOCK.
```

```
   IF FB-NOLOCK THEN PERFORM FCT-READIND-NOLOCK.

FCT-REWRITE.
   MOVE ENREG  TO REQ-DATA.
   MOVE RECLON TO UERQDATAL.
   MOVE RECLON TO UERRDATAL.
   PERFORM FBSS-RP.

FCT-DELETE.
   PERFORM FBSS-DL.

FCT-WRITE.
   MOVE ENREG  TO REQ-DATA.
   MOVE RECLON TO UERQDATAL.
   MOVE RECLON TO UERRDATAL.
   PERFORM FBSS-IS.

FCT-BEGTRANS.
   PERFORM FBSS-BT.

FCT-ENDTRANS.
   PERFORM FBSS-ET.

FCT-READFIRST-LOCK.
   MOVE "GE" TO OPERATOR.
   PERFORM FBSS-HU.

FCT-READFIRST-NOLOCK.
   MOVE "GE" TO OPERATOR.
   PERFORM FBSS-GU.


FCT-READIND-LOCK.
   MOVE "EQ" TO OPERATOR.
   PERFORM FBSS-HU.


FCT-READIND-NOLOCK.
   MOVE "EQ" TO OPERATOR.
   PERFORM FBSS-GU.

FCT-READSTEP-LOCK.
   MOVE "GT" TO OPERATOR.
   PERFORM FBSS-HU.

FCT-READSTEP-NOLOCK.
   MOVE "GT" TO OPERATOR.
   PERFORM FBSS-GU.

ANALYSE-RET-CODE-SECTION.
   IF     UERRETCODE EQUAL TO LOW-VALUE
      AND UERSERVRC  EQUAL TO LOW-VALUE
         THEN MOVE 0 TO RETCOD.
   IF UERRETCODE NOT EQUAL TO LOW-VALUE
```

```
           THEN PERFORM ANAL-UERRETCODE-SECTION.
      IF UERSERVRC  NOT EQUAL TO LOW-VALUE
           THEN PERFORM ANAL-UERSERVRC-SECTION.

   ANAL-UERRETCODE-SECTION.
      MOVE 16 TO RETCOD.
      MOVE UERRETCODE TO VARRET.
      EXIT PROGRAM.

   ANAL-UERSERVRC-SECTION.
      MOVE 16 TO RETCOD.
      MOVE UERSERVRC TO VARRET.
      IF       VARRET   = "EG"
         AND   FUNCTION = "RS"
         THEN MOVE "EF" TO DEB
              MOVE  8 TO RETCOD.
      IF       VARRET   = "EG"
         AND   FUNCTION = "RF"
         THEN MOVE "EF" TO DEB
              MOVE  8 TO RETCOD.
      IF       VARRET   = "EG"
         AND   FUNCTION = "RI"
         THEN MOVE "NF" TO DEB
              MOVE  4 TO RETCOD.
      IF VARRET = "DA" THEN MOVE   4  TO RETCOD.
      IF VARRET = "AD" THEN MOVE  16  TO RETCOD
                            MOVE "IR" TO DEB.
      IF VARRET = "JD" THEN MOVE "UN" TO DEB
                            MOVE   4  TO RETCOD.
      IF VARRET = "BG" THEN MOVE "EF" TO DEB
                            MOVE   8  TO RETCOD.
      IF VARRET = "II" THEN MOVE "DR" TO DEB
                            MOVE   4  TO RETCOD.
      IF VARRET = "LR" THEN MOVE "LO" TO DEB
                            MOVE  12  TO RETCOD.
      EXIT PROGRAM.


   COPY FBSS-FCT.CBL.


   ***************************************************************
   *                        FIN DU FICHIER FIC.CBL                *
   ***************************************************************
   ***************************************************************
```

2.2.2 Zone d'interface entre les applications ICE et ICEFIC

```
   ***************************************************************
   * Fichier : WFICINTR.CBL
   *
   * Contenu : Ce fichier contient la déclarationde l'interface de FIC.
   *
```

```
* Auteur : D. Bauchau.
*
* Date : 17 octobre 89.
*
*****************************************************************


 01 FUNCTION   PIC X(2).

     88 FB-READFIRST; VALUE "RF".
     88 FB-READSTEP ; VALUE "RS".
     88 FB-READIND  ; VALUE "RI".
     88 FB-REWRITE  ; VALUE "RW".
     88 FB-DELETE   ; VALUE "DL".
     88 FB-WRITE    ; VALUE "WR".
     88 FB-BEGTRANS ; VALUE "BT".
     88 FB-ENDTRANS ; VALUE "ET".


 01 PARAM      PIC X(2).

     88 FB-LOCK     ; VALUE "LO".
     88 FB-NOLOCK   ; VALUE "NL".

 01 DEB PIC X(2).

 01 FICH.

     05 ENTETE.

          10 RETCOD PIC S9(008) COMP.
          10 NOMFIC PIC  X(006).
          10 FILLER PIC  X(001).
          10 NUMCLE PIC  X(001).
          10 LONENR PIC S9(008) COMP.

     05 ENREG  PIC X(1024).
```

2.2.3 Listing du programme COBOL RPT

```
*****************************************************************
* Fichier : RPT.CBL
*
* Contenu : Ce programme permet de réaliser un accès au répertoire
*
*           des fichiers FBSS.
*
* Auteur : D. Bauchau.
*
* Date : 5 octobre 89.
*
*****************************************************************
```

```
*-----------------------------------------------------------------*
 IDENTIFICATION DIVISION.
*-----------------------------------------------------------------*
   PROGRAM-ID. RPT.
   AUTHOR. D-BAUCHAU.



*-----------------------------------------------------------------*
 DATA DIVISION.
*-----------------------------------------------------------------*

*.................................................................*
 WORKING-STORAGE SECTION.
*.................................................................*

 COPY WDB-INIT.CBL.

 copy wsvpsais.cbl.

*.................................................................*
 LINKAGE SECTION.
*.................................................................*

 COPY WRPTSTRC.CBL.


 01 ERREUR PIC 9(1).

*.................................................................*
 screen SECTION.
*.................................................................*

 COPY ssvpsais.CBL.
 copy ssvpaffr.cbl.



*-----------------------------------------------------------------*
 PROCEDURE DIVISION USING ENREGRPT, ERREUR.
*-----------------------------------------------------------------*

 DEBUT-RPT.

      PERFORM FBSS-INIT-CPRB-DB.
      MOVE NOMLOGFI   TO REQ-DATA.
      MOVE "PCBRPRTR" TO PCBNAME.
      PERFORM FBSS-GU.
      MOVE REP-DATA TO ENREGRPT.
      MOVE 1 TO ERREUR.
      IF UERRETCODE = LOW-VALUE AND UERSERVRC = LOW-VALUE
            THEN MOVE 0 TO ERREUR.
```

```
EXIT PROGRAM.

FBSS-BT.
   MOVE "TB" TO UERFUNCT.
   MOVE "        " TO PCBNAME OF REQ-PARM.
   MOVE "  " TO OPERATOR OF REQ-PARM.
   MOVE 0 TO UERQDATAL UERRDATAL.
   CALL "_SEND_REQUEST" USING UERCPRB.


FBSS-GU.
   MOVE "UG" TO UERFUNCT.
   MOVE 256  TO UERRDATAL.
   MOVE "EQ" TO OPERATOR.
   MOVE 6    TO UERQDATAL.
   CALL "_SEND_REQUEST" USING UERCPRB.

FBSS-ET.
   MOVE "TE" TO UERFUNCT.
   MOVE "        " TO PCBNAME OF REQ-PARM.
   MOVE "  " TO OPERATOR OF REQ-PARM.
   MOVE 0 TO UERQDATAL UERRDATAL.
   CALL "_SEND_REQUEST" USING UERCPRB.

FBSS-INIT-CPRB-DB.
   SET UERSERVER  TO ADDRESS OF DBSERVEUR.
   SET UERQPARMAD TO ADDRESS OF REQPARMAREA.
   SET UERQDATAAD TO ADDRESS OF REQ-DATA.
   SET UERRPARMAD TO ADDRESS OF REPPARMAREA.
   SET UERRDATAAD TO ADDRESS OF REP-DATA.


*****************************************************************
*                        FIN DU FICHIER RPT                     *
*****************************************************************
*****************************************************************
```

2.2.4 WORKING STORAGE SECTION du programme RPT

```
*****************************************************************
* Fichier : WRPTSTRC.CBL
*
* Contenu : Ce programme contient la structure du répertoire.
*
* Auteur : D. Bauchau.
*
* Date : 5 OCTOBRE 89.
*
*****************************************************************

 01 ENREGRPT.
```

```
           05 NOMLOGFI PIC X(6).

           05 NOM-SRVR PIC X(8).

           05 NOMCLE   PIC 9(1).

           05 RECLON   PIC 9(4).

           05 TABLEAU.
                10 ELEMENT OCCURS 6 INDEXED BY INDICE.
                     20 LONGUEUR PIC X(4).
                     20 POSIT    PIC X(4).

           05 NOMPHYFI PIC X(12).

           05 FILLER   PIC X(182).
```

2.2.5 Listing du programme EXT en COBOL

```
      ******************************************************************
      * Fichier : EXT.CBL
      *
      * Contenu : Ce programme permet d'extraire d'un string "DONNEE",
      *           de longueur 1024
      *           un string de position 0 < "POSSTR" < 1025
      *                  et de longueur 0 < "LONSTR" < 1025
      *           et de placer le resultat cadré à gauche
      *           dans une variable "RESULT" de longueur 1024
      *
      * Auteur : D. Bauchau.
      *
      * Date : 19 octobre 89.
      *
      ******************************************************************


      *-----------------------------------------------------------------*
       IDENTIFICATION DIVISION.
      *-----------------------------------------------------------------*
          PROGRAM-ID. EXT.
          AUTHOR. D-BAUCHAU.



      *-----------------------------------------------------------------*
       DATA DIVISION.
      *-----------------------------------------------------------------*

      *.................................................................*
       WORKING-STORAGE SECTION.
      *.................................................................*
```

```
 01 VAL001 PIC 9(4).

 01 VAL002 PIC 9(4).


*.....................................................................*
 LINKAGE SECTION.
*.....................................................................*

 COPY LEXTSTRC.CBL.


*---------------------------------------------------------------------*
 PROCEDURE DIVISION USING LEXTSTRC.
*---------------------------------------------------------------------*

 DEBUT-EXT.
   MOVE POSSTR TO VAL001.
   MOVE 1      TO VAL002.
   PERFORM BOUCLE-SECTION LONSTR TIMES.
   EXIT PROGRAM.


 BOUCLE-SECTION.
    SET INDDON TO VAL001.
    SET INDRES TO VAL002.
    IF ( VAL001 < 1025 ) AND ( VAL002 < 1025 )
          THEN MOVE ELEDON ( INDDON ) TO ELERES ( INDRES ).
    COMPUTE VAL001 = VAL001 + 1.
    COMPUTE VAL002 = VAL002 + 1.


**********************************************************************
*                          FIN DU FICHIER EXT.CBL                    *
**********************************************************************
**********************************************************************
```

2.2.6 Listing de la LINKAGE SECTION du programme EXT

```
**********************************************************************
* Fichier : LEXTSTRC.CBL
*
* Contenu : Ce programme contient la LS pour la fonction EXT
*
**********************************************************************

 01 LEXTSTRC.
       02 DONNEE.
            05 ELEDON PIC X(1) OCCURS 1024 INDEXED BY INDDON.
```

```
02 POSSTR PIC 9(4).
02 LONSTR PIC 9(4).
02 RESULT.
     05 BIDRES PIC X(1024) VALUE SPACES.
     05 BBBRES REDEFINES BIDRES.
          07 ELERES PIC X(1)
                OCCURS 1024 INDEXED BY INDRES.
```

2.3 Le précompilateur ICE

2.3.1 Listing du précompilateur ICE en PASCAL

```
{*****************************************************************************}
{     Fichier : ICE.PAS                                                       }
{                                                                             }
{     Contenu : Ce fichier contient le texte du précompilateur ICE-PC         }
{                                                                             }
{     Auteur : D. Bauchau.                                                    }
{                                                                             }
{     Date : 28 Novembre 89.                                                  }
{                                                                             }
{*****************************************************************************}




{*****************************************************************************}
  PROGRAM ICE;
{*****************************************************************************}


{*****************************************************************************}
{ DECLARARTION DES FICHIERS INCLUS ET AUTRES INTERFACES                        }
{*****************************************************************************}

  USES CRT,DOS,TURBO3;


{*****************************************************************************}
{ DECLARARTION DES CONSTANTES                                                 }
{*****************************************************************************}

  CONST
     LLIGNE=100;
     const000='|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||
||||||||||||';
     MARGE06='      ';
     MARGE07='       ';
     MARGE10='          ';
     MARGE11='           ';
     MARGE15='               ';
     MARGE20='                    ';


{*****************************************************************************}
{ DECLARARTION DE TYPE                                                        }
{*****************************************************************************}

  TYPE

     FOT = TEXT;
```

```
    TLIGNE = STRING[LLIGNE];
    TTLIGNE = (ERR,COM,LAB,D01,INSTR,VID,AUT);


{*****************************************************************************}
{ DECLARARTION DE VARIABLE                                                    }
{*****************************************************************************}

  VAR

   {DECLARATION DES FICHIERS}
     SOURCE,                 {le fichier SOURCE}
     TARGET,                 {le fichier TARGET}
     MESSAGE,                {le fichier des lignes cobol à rajouter dans TARGET}
     ERREUR   :FOT;          {le fichier des messages d'erreurs}

   {DECLARATION DES VARIABLES GLOBALES}
     LIGNE,                  {la ligne courante lue dans SOURCE}

     NFSOUR,                 {le nom système du fichier SOURCE}
     NFTARG: TLIGNE;         {le nom système du fichier TARGET}
     TYPE_LIGNE:TTLIGNE;     {le type de la ligne courante}


     R_ENV_DIV,              {VRAI si on a rajouter l'ENVIRONMENT DIVISION}
     R_DATA_DIV,             {VRAI si on a rajouter la DATA DIVISION}
     R_ICEITR,               {VRAI si on a rajouter dans ICEITR}
     R_ICEIPR,               {VRAI si on a rajouter dans ICEIPR}
     R_ICETTR,               {VRAI si on a rajouter dans ICETTR}
     R_ICETPR : Boolean;     {VRAI si on a rajouter dans ICETPR}


     cligne : integer;       {le numéro de la ligne courante du fichier SOURCE}

     poscurx,                {le numéro de la ligne du curseur}
     poscury : integer;      {le numéro de la colone du curseur}

     COMPTEUR_ECRAN:integer;{le nombre d'écran déja rencontré}

   {DECLARATION DES VARIABLES DE DIRECTIVES DE (PRE)COMPILATION}
     DC0001:Boolean;         {vrai s'il faut supprimer les commentaires}

   {DECLARATION DES VARIABLES SYSTEME}
     Year,                   {l'année système}
     Month,                  {le moi système}
     Day,                    {le jour système}
     DayOfWeek,              {le jour de la semaine système}
     Hour,                   {l'heure système}
     Minute,                 {la minute système}
     Second,                 {la seconde système}
     Sec100 : Word;          {les centièmes de seconde système}


  {-----------------------------------------------------------------------}
```

```
 FUNCTION SSBLANC(STR:TLIGNE):TLIGNE;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                               }
{   Retire tout les blancs                                                      }
{-------------------------------------------------------------------------------}

   VAR
     i:integer;
     l:tligne;

   BEGIN
     l:='';
     for i:=1 to length(str) do if str[i]<>' ' then l:=l+str[i];
     ssblanc:=l
   END;



{-------------------------------------------------------------------------------}
 FUNCTION upper(STR:TLIGNE):TLIGNE;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                               }
{   Met en maguscule                                                            }
{-------------------------------------------------------------------------------}

   VAR
     i:integer;
     l:tligne;

   BEGIN
     l:='';
     for i:=1 to length(str) do l:=l+upcase(str[i]);
     upper:=l;
   END;


{-------------------------------------------------------------------------------}
 FUNCTION LTRIM(STR:TLIGNE):TLIGNE;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                               }
{   Retire les blancs de gauche s'il y en a                                     }
{-------------------------------------------------------------------------------}

   VAR
     I,P:integer;

   BEGIN
     I:=0;
     P:=0;
     WHILE I<LENGTH(STR) DO
       BEGIN
         I:=I+1;
         IF STR[I]<>' '
              THEN {on a le premier caractère différent de space}
```

```
                BEGIN
                  P:=I;
                  I:=LENGTH(STR);
                END;
        END;
      LTRIM:=COPY(STR,P,LENGTH(STR));
    END;

{-------------------------------------------------------------------------------}
 FUNCTION REV(STR:TLIGNE):TLIGNE;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                               }
{   Renverse un string STR                                                      }
{-------------------------------------------------------------------------------}
    VAR i:integer;
        rts:tligne;

    BEGIN
      rts:=str;
      for i:=1 to length(str) do rts[length(str)-i+1]:=str[i];
      rev:=rts
    END;




{-------------------------------------------------------------------------------}
 FUNCTION RTRIM(STR:TLIGNE):TLIGNE;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                               }
{   Retire les blancs de droite s'il y en a                                     }
{-------------------------------------------------------------------------------}

    VAR
      RTS:TLIGNE;

    BEGIN
      RTS:=REV(STR);
      RTS:=LTRIM(RTS);
      RTRIM:=REV(RTS);
    END;


{-------------------------------------------------------------------------------}
 PROCEDURE trans_int_str_3(int:integer; VAR st:TLIGNE);
{-------------------------------------------------------------------------------}
{SPECIFICATIONS : transforme un entier en un string de longeur 3 précédé de    }
{     zéro si nécessaire ex: 22 => 022                                          }
{-------------------------------------------------------------------------------}

    var ST01 : TLIGNE;
```

```
   BEGIN
     STR(INT,ST01);;
     IF (INT<999) AND (INT>0)
           THEN
             BEGIN
                IF (INT>99)              THEN ST:=ST01;
                IF (INT>9) AND (INT<100) THEN ST:='0'+ST01;
                IF (INT<10)              THEN ST:='00'+ST01;
           END;
   END;


{-------------------------------------------------------------------------}
 FUNCTION Check_File(VAR F:Fot):Boolean;
{-------------------------------------------------------------------------}
{SPECIFICATIONS : Vrai si le fichier "F" existe et Fauwx si non.          }
{-------------------------------------------------------------------------}

   BEGIN
     {$I-}
     reset(f);
     {$I-}
     IF IoResult=0 THEN Check_File:=TRUE
                   ELSE Check_File:=FALSE;
   END;


{-------------------------------------------------------------------------}
 PROCEDURE ICE_CLOT;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   Fermeture de tout les fichiers ouvert                                 }
{-------------------------------------------------------------------------}


   BEGIN;
      CLOSE(SOURCE);
      CLOSE(TARGET);
      CLOSE(MESSAGE);
      CLOSE(ERREUR);
      writeln;
   END;


{-------------------------------------------------------------------------}
 PROCEDURE POSIT(N:tligne ; VAR F:fot);
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   On se Positionne sur l'enregistrement numéro "N" dans le fichier "F"  }
{-------------------------------------------------------------------------}


   VAR
     l,b,v:tligne;
```

```
   BEGIN
     reset(f);
     b:='|NUM='+n;
     readln(f,l);
     WHILE (l<>b) AND (not eof(f)) DO readln(f,l);
   END;

{-------------------------------------------------------------------------}
 FUNCTION CARI(N:INTEGER):tligne;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   Renvoie une chaine de caractères equivalente à l'entier "N"           }
{-------------------------------------------------------------------------}

   VAR
      v : tligne;

   BEGIN
      str(n:2,v);
      cari:=v;
   END;


{-------------------------------------------------------------------------}
 PROCEDURE INS(N:tligne; var f1:fot; var f2:fot);
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   Ecriture du message numéro N du fichier FS dans le fichier F2.        }
{-------------------------------------------------------------------------}

   VAR
      l:tligne;

   BEGIN
      posit(n,f1);
      readln(f1,l);
      WHILE (l<>const000) AND (not eof(f1)) DO
        BEGIN
          writeln(f2,l);
          readln(f1,l);
        END;
   END;


{-------------------------------------------------------------------------}
 PROCEDURE Stop_Erreur(NumErr:Tligne);
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   On a découvert une erreur On l'affiche On ferme tout et on arrete     }
{-------------------------------------------------------------------------}

   BEGIN
     writeln;
```

```
    writeln('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ');
    writeln('- ',Cligne,' -',Ligne);
    writeln('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ');
    INS(NumErr,Erreur,Output);
    ICE_CLOT;
    HALT;
  END;



{-------------------------------------------------------------------------------}
 PROCEDURE PUT_ENTETE;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                               }
{   Ecriture de l'entête dans le fichier TARGET                                 }
{-------------------------------------------------------------------------------}

  VAR
     l:tligne;

  BEGIN
     l:='      ****************************************************************';
     writeln(target,l);
     l:='      * FICHIER REALISE PAR PRECOMPILATION DU FICHIER ';
     l:=l+upper(NfSour)+'                  ';
     l[71]:='*';
     writeln(target,l);
     l:='      ****************************************************************';
     writeln(target,l);
     l:='      * PRECOMPILE PAR ICE VERSION 0.0 NOVEMBRE 89                   *';
     writeln(target,l);
     l:='      ****************************************************************';
     writeln(target,l);
     GetDate(Year,Month,Day,DayOfWeek);
     GetTime(Hour,Minute,Second,Sec100);
     l:='      * LE '+cari(day)+'/'+cari(month)+'/'+cari(year);
     l:=l+'                                    *';
     writeln(target,l);
     l:='      ****************************************************************';
     writeln(target,l);
     l:='      * A  '+cari(hour)+':'+cari(minute)+':'+cari(second)+':'+cari(sec100);
     l:=l+'                                  *';
     writeln(target,l);
     l:='      ****************************************************************';
     writeln(target,l);
     writeln(target);
     writeln(target);
     writeln(target);

  END;
```

```
{-------------------------------------------------------------------------}
 PROCEDURE DISPLAY_ENTETE;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   Affiche à l'écran le message de départ du programme et saufgarde les  }
{   coordonnée du curseur afin de pouvoir afficher le numéro de la ligne  }
{   traitée                                                               }
{-------------------------------------------------------------------------}

   BEGIN
      writeln;
      writeln('  PRECOMPILATEUR ICE VERSION 0.0 Nov 89. ');

   END;

{-------------------------------------------------------------------------}
 PROCEDURE FILES_INITIALISATION_01;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   Assignation et initialisation des fichiers nécessaires aux premières  }
{   vérification                                                          }
{-------------------------------------------------------------------------}

   BEGIN
      ASSIGN(MESSAGE,'MESSAGE.ICE');
      IF Check_File(Message)
             THEN
               RESET(MESSAGE)
             ELSE
               BEGIN
                 writeln;
                 writeln;
                 writeln('    Fichier MESSAGE.ICE introuvable !');
                 Ice_Clot;
                 Halt;
               END;
      ASSIGN(ERREUR,'ERREUR.ICE');
      IF Check_File(Erreur)
             THEN
               RESET(Erreur)
             ELSE
               BEGIN
                 writeln;
                 writeln;
                 writeln('    Fichier ERREUR.ICE introuvable !');
                 Ice_Clot;
                 Halt;
               END;
   END;


{-------------------------------------------------------------------------}
 PROCEDURE FILES_INITIALISATION_02;
{-------------------------------------------------------------------------}
```

```
{SPECIFICATIONS :                                                              }
{   Assignation et initialisation des autres fichiers                          }
{------------------------------------------------------------------------------}

   BEGIN
      ASSIGN(SOURCE,NFSOUR);
      IF Check_File(SOURCE)
             THEN
               RESET(SOURCE)
             ELSE
               BEGIN
                 writeln;
                 writeln;
                 writeln('    Fichier ',NFSOUR,' introuvable !');
                 Ice_Clot;
                 Halt;
               END;
      ASSIGN(TARGET,NFTARG);
      REWRITE(TARGET);
   END;



{------------------------------------------------------------------------------}
 PROCEDURE VARIABLES_INITIALISATION;
{------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                              }
{   Initialisation de toutes les variables globales a initialiser              }
{------------------------------------------------------------------------------}

   BEGIN
      DC0001 :=false;   {à priori il ne faut pas supprimer les commentaires}
      COMPTEUR_ECRAN := 0;
      R_ENV_DIV :=FALSE;
      R_DATA_DIV:=FALSE;
      R_ICEITR  :=FALSE;
      R_ICETTR  :=FALSE;
      R_ICETPR  :=FALSE;
      R_ICEIPR  :=FALSE;
   END;


{------------------------------------------------------------------------------}
 PROCEDURE CHECK_PARAM;
{------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                              }
{   Vérifie qu'il y a bien 2 paramètres et que le premier est bien le nom      }
{   d'un fichier existant dans le répertoire courant                           }
{------------------------------------------------------------------------------}

   VAR
      VAR001:char;      {'V' si les commentaires sont à supprimer, 'F' sinon}
      VARNOM:TLigne;
      SX,SY:INTEGER;
```

```
BEGIN
   NfSour:='';
   NfTarg:='';
   CASE PARAMCOUNT OF
      0 : BEGIN
           INS('0001E',ERREUR,OUTPUT);
           SX:=WhereX;
           SY:=Wherey;
           write('  Donnez le nom du fichier SOURCE ICE à précompiler › ');
           readln(NFSOUR);
           IF pos('.',NfSour)=0
                  THEN {il n'y a pas d'extension}
                    BEGIN
                      NfSour:=NfSour+'.ICE';
                      GotoXY(SX,SY);
                      writeLn('  Donnez le nom du fichier SOURCE ICE à préco
piler › ',NfSour);
                    END;
           {le nom par défaut du fichier target est}
           VARNOM:=copy(NfSour,1,pos('.',NfSour)-1)+'.CBL';
           write('  Donnez le nom du fichier TARGET CBL à construire  › ');
           SX:=WhereX;
           SY:=WhereY;
           write(VARNOM);
           GotoXY(SX,SY);
           readln(NFTARG);
           VAR001:='F';
           WRITE('  Les commentaires sont-ils à supprimer (Vrai/Faux) › ', VA
001);
           GOTOXY(WHEREX-1,WHEREY);
           READ(KBD,VAR001);
           VAR001:=UPCASE(VAR001);
           IF VAR001='V'
                THEN DC0001:=TRUE
                ELSE DC0001:=FALSE;
      END;
      1: BEGIN
           Nfsour:=paramstr(1);
           IF pos('.',NfSour)=0
                  THEN {il n'y a pas d'extension}
                       NfSour:=NfSour+'.ICE';
           writeLn('  Fichier SOURCE ICE à précompiler › ',NfSour);
           {le nom par défaut du fichier target est}
           NfTarg:=copy(NfSour,1,pos('.',NfSour)-1)+'.CBL';
           write('  Fichier TARGET CBL à construire  › ',Nftarg);
           DC0001:=FALSE;
      END;
      2: BEGIN
           NFSOUR:=PARAMSTR(1);
           NFTARG:=PARAMSTR(2);
           DC0001:=FALSE;
      END;
      3: BEGIN
           NFSOUR:=PARAMSTR(1);
```

```
            NFTARG:=PARAMSTR(2);
            IF PARAMSTR(3)='sans*' THEN DC0001:=TRUE;
          END;
    END;{endcase paramcount}
    IF pos('.',NfSour)=0 THEN NfSour:=NfSour+'.ICE';
    IF pos('.',NfSour)>0 THEN VARNOM:=copy(NfSour,1,pos('.',NfSour)-1);
    IF NfTarg=''         THEN NfTarg:=VARNOM+'.CBL';
    IF pos('.',NfTarg)=0 THEN NfTarg:=NfTarg+'.CBL';

    writeln;
    writeln;
    write  ('  Ligne en cours de traîtement : ');
    poscurx:=wherex;
    poscury:=wherey;
  END;
```

```
{-------------------------------------------------------------------------}
 PROCEDURE ICE_INIT;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   Réalise : l'affichage de l'entête,                                    }
{             l'assignation des fichiers,                                 }
{             la vérification des paramètres de la ligne de commande,     }
{             l'initialisation des variables globale du programme,        }
{             insère au début du fichier target une série de commentaires }
{-------------------------------------------------------------------------}


  BEGIN
    CLRSCR;
    DISPLAY_ENTETE;
    VARIABLES_INITIALISATION;
    FILES_INITIALISATION_01;
    CHECK_PARAM;
    FILES_INITIALISATION_02;
    PUT_ENTETE;
  END;
```

```
{-------------------------------------------------------------------------}
 PROCEDURE RAJOUTE( MC:TLIGNE;
                    TMC:TTLIGNE;
                    NUM:TLIGNE;
                    RAJ_PREC:Boolean;
               VAR  RAJ_COUR:Boolean);
{-------------------------------------------------------------------------}
```

```
{SPECIFICATIONS : Si on rencontre une ligne contenant le mot clef MC de type }
{        TMC que RAJ_PREC est vrai et que RAJ_COUR est faux alors on rajoute }
{        le message numéro MUM du                                            }
{        fichier MESSAGE.ICE dans le fichier TARGET avant la ligne courante  }
{        qui contient le mot clef et l'on donne à SENT la valeur VRAI        }
{-------------------------------------------------------------------------}

   VAR
     VARLGN:tligne;
     VARPOS:integer;

   BEGIN
      VarLgn:=upper(ssblanc(ligne));
      VarPos:=POS(MC,varlgn);

      IF (VarPos<>0)
         AND ( not RAJ_PREC )
         AND ( Type_Ligne=TMC )
           THEN Stop_Erreur('0009E');

      IF (VarPos<>0)
         AND ( RAJ_PREC )
         AND ( NOT RAJ_COUR )
         AND ( Type_Ligne=TMC )
           THEN
             BEGIN
               INS(NUM,MESSAGE,TARGET);
               RAJ_COUR:=true;
             END;
   END;


{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ENV_DIV;
{-------------------------------------------------------------------------}
{SPECIFICATIONS : Si on rencontre une ligne avec 'DATA DIVISION'             }
{      alors rajouter juste avant cette ligne ce que ICE doit rajouter à    }
{      cet endroit.                                                         }
{-------------------------------------------------------------------------}


   BEGIN
      RAJOUTE('DATADIVISION.', LAB, '0002A', TRUE, R_ENV_DIV);
   END;



{-------------------------------------------------------------------------}
 PROCEDURE RAJ_DATA_DIV;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :   Si on rencontre une ligne avec 'SCREEN SECTION'          }
{      alors rajouter juste avant cette ligne ce que ICE doit rajouter à    }
{      cet endroit et signaler que l'on a déja fait cet ajout.              }
{-------------------------------------------------------------------------}
```

```
   BEGIN
      RAJOUTE('SCREENSECTION.', LAB, '0003A', R_ENV_DIV, R_DATA_DIV);
   END;


{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ICEIPR;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :  Si on rencontre une ligne 'ICEIPR                       }
{      alors rajouter juste avant cette ligne ce que ICE doit rajouter à   }
{      cet endroit et signaler que l'on a déja fait cet ajout.             }
{-------------------------------------------------------------------------}


   BEGIN
      RAJOUTE('ICEIPR.', LAB, '0011A', R_DATA_DIV, R_ICEIPR);
   END;



{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ICEITR;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :  Si on rencontre une ligne 'ICE001-I                     }
{      alors rajouter juste avant cette ligne ce que ICE doit rajouter à   }
{      cet endroit et signaler que l'on a déja fait cet ajout.             }
{-------------------------------------------------------------------------}


   BEGIN
      RAJOUTE('ICE001-I.', LAB, '0005A', R_DATA_DIV, R_ICEITR);
   END;



{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ICEXXX_I;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :  Si on rencontre une ligne 'ICEXXX-T                     }
{        alors rajouter juste avant cette ligne ce que ICE doit rajouter à }
{        cet endroit.                                                      }
{-------------------------------------------------------------------------}


   VAR ST01,ST02,ST03,varlgn:tligne;



   BEGIN
        TRANS_INT_STR_3(COMPTEUR_ECRAN+1,ST03);
```

```
        ST01:='ICE'+ST03+'-T.';
        varlgn:=upper(ssblanc(ligne));
        IF (POS(st01,varlgn)<>0)
              THEN
                BEGIN
                  COMPTEUR_ECRAN:=COMPTEUR_ECRAN+1;
                  TRANS_INT_STR_3(COMPTEUR_ECRAN,ST03);
                  ST02:=MARGE10+'MOVE '+ST03+'  TO ICECUR.';
                  WRITELN(TARGET,ST02);
                  ST02:=MARGE10+'PERFORM ICEAFF.';
                  WRITELN(TARGET,ST02);
                END
   END;

{-------------------------------------------------------------------------------}
 PROCEDURE COMPTE_ECRAN;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :  Maitient à jour la variable COMPTEUR_ECRAN                   }
{-------------------------------------------------------------------------------}


   VAR ST01,ST02,ST03,varlgn:tligne;



   BEGIN
        TRANS_INT_STR_3(COMPTEUR_ECRAN+1,ST03);
        ST01:='ICE'+ST03+'-I.';
        varlgn:=upper(ssblanc(ligne));
        IF (POS(st01,varlgn)<>0) THEN COMPTEUR_ECRAN:=COMPTEUR_ECRAN+1;

   END;


{-------------------------------------------------------------------------------}
 PROCEDURE RAJ_ICEXXX_T;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :  Si on rencontre une ligne 'ICEXXX-I                          }
{        alors rajouter juste avant cette ligne ce que ICE doit rajouter à      }
{        cet endroit.                                                           }
{-------------------------------------------------------------------------------}


   VAR ST01,ST02,ST03,varlgn:tligne;



   BEGIN
        TRANS_INT_STR_3(COMPTEUR_ECRAN+1,ST03);
        ST01:='ICE'+ST03+'-I.';
        varlgn:=upper(ssblanc(ligne));
        IF (POS(st01,varlgn)<>0) OR (POS('ICETTR.',varlgn)<>0)
```

```
                THEN
                  BEGIN
                    TRANS_INT_STR_3(COMPTEUR_ECRAN+1,ST03);
                    ST02:=MARGE11+'IF ICEETR = ''Y''';
                    WRITELN(TARGET,ST02);
                    ST02:=MARGE11+'                PERFORM FBSS-ET';
                    WRITELN(TARGET,ST02);
                    ST02:=MARGE11+'                GO TO ICETPR.';
                    WRITELN(TARGET,ST02);
                    ST02:=MARGE11+'IF ICEATR = ''Y''';
                    WRITELN(TARGET,ST02);
                    ST02:=MARGE11+'                PERFORM FBSS-RB';
                    WRITELN(TARGET,ST02);
                    ST02:=MARGE11+'                GO TO ICEITR.';
                    WRITELN(TARGET,ST02);
                  END
    END;


{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ICETTR;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :  Si on rencontre une ligne 'ICETPR'                      }
{      alors rajouter juste avant cette ligne ce que ICE doit rajouter à   }
{      cet endroit et signaler que l'on a déja fait cet ajout.             }
{-------------------------------------------------------------------------}

    BEGIN
       RAJOUTE('ICETPR.', LAB, '0007A', R_ICEITR, R_ICETTR);
    END;


{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ICETPR;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :  Si on rencontre une ligne un label apres ICETPR         }
{   alors rajouter juste avant le prochain label ou avant la fin du fichier }
{   ce que ICE doit rajouter à cet endroit et signaler que l'on a déja     }
{    fait cet ajout.                                                       }
{-------------------------------------------------------------------------}



    BEGIN
       IF R_ICETTR THEN RAJOUTE('.', LAB, '0008A', R_ICETTR, R_ICETPR);
    END;


{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ICEAFF;
{-------------------------------------------------------------------------}
{SPECIFICATIONS : Construit et rajoute dans TARGET le paragraphe ICEAFF    }
{       dans la ZONE DES PERFORM                                           }
{-------------------------------------------------------------------------}
```

```
VAR
  ST01 : INTEGER;
  ST02,ST03,ST04 : TLIGNE;


BEGIN
  INS('0009A',MESSAGE,TARGET);
  ST02:=MARGE07+'ICEAFFSCR.';
  WRITELN(TARGET,ST02);
  FOR ST01:=1 TO COMPTEUR_ECRAN DO
        BEGIN
          TRANS_INT_STR_3(ST01,ST03);
          ST02:=MARGE10+'IF ICECUR = '+ST03;
          WRITELN(TARGET,ST02);
          ST02:=MARGE15+'DISPLAY ICE'+ST03;
          WRITELN(TARGET,ST02);
          ST02:=MARGE15+'IF ICEPF1 = ''Y''';
          WRITELN(TARGET,ST02);
          ST02:=MARGE20+'DISPLAY ''PF1 pour confirmer''';
          WRITELN(TARGET,ST02);
          ST02:=MARGE20+'AT LINE 24 COL 1';
          WRITELN(TARGET,ST02);
          ST02:=MARGE20+'WITH REVERSE-VIDEO.';
          WRITELN(TARGET,ST02);
          ST02:=MARGE15+'ACCEPT  ICE'+ST03+'.';
          WRITELN(TARGET,ST02);
        END;
  INS('0010A',MESSAGE,TARGET);
  { on peut commencer à insérer ici }

  ST04:='';
  FOR ST01:=1 TO COMPTEUR_ECRAN DO
        BEGIN
          TRANS_INT_STR_3(ST01,ST03);
          ST04:=ST04+'WICE'+ST03;
          IF ST01 < COMPTEUR_ECRAN THEN ST04:=ST04+', ';
        END;
  ST02:=MARGE20+'GO TO '+ST04+' DEPENDING ON ICENXT';
  WRITELN(TARGET,ST02);
  ST02:=MARGE11+'ELSE     GO TO '+ST04+' DEPENDING ON ICECUR.';
  WRITELN(TARGET,ST02);
  INS('0012A',MESSAGE,TARGET);
  FOR ST01:=1 TO COMPTEUR_ECRAN DO
        BEGIN
          TRANS_INT_STR_3(ST01,ST03);
          ST02:=MARGE07+'WICE'+ST03+'.';
          WRITELN(TARGET,ST02);
          ST02:=MARGE11+'IF ICECCC = ''I''';
          WRITELN(TARGET,ST02);
          ST02:=MARGE20+'MOVE '+ST03+' TO ICECUR';
          WRITELN(TARGET,ST02);
          ST02:=MARGE20+'MOVE ''T'' TO ICECCC';
          WRITELN(TARGET,ST02);
```

```
            ST02:=MARGE20+'MOVE ''N'' TO ICEREF';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'GO TO ICE'+ST03+'-I.';
            WRITELN(TARGET,ST02);

            ST02:=MARGE11+'IF ICECCC = ''T''';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE '+ST03+' TO ICECUR';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE ''D'' TO ICECCC';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'GO TO ICEAFF.';
            WRITELN(TARGET,ST02);

            ST02:=MARGE11+'IF ICECCC = ''D''';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE '+ST03+' TO ICECUR';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE ''E'' TO ICECCC';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'GO TO ICE'+ST03+'-T.';
            WRITELN(TARGET,ST02);

            ST02:=MARGE11+'IF ICECCC = ''E''';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE '+ST03+' TO ICECUR';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE ''F'' TO ICECCC';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'GO TO ICE'+ST03+'-E.';
            WRITELN(TARGET,ST02);

            ST02:=MARGE11+'IF ICECCC = ''F''';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE '+ST03+' TO ICECUR';
            WRITELN(TARGET,ST02);
            ST02:=MARGE20+'MOVE ''I'' TO ICECCC.';
            WRITELN(TARGET,ST02);


            ST02:=MARGE11+'GO TO ICE.';
            WRITELN(TARGET,ST02);
          END;


   END;

{-------------------------------------------------------------------------}
 PROCEDURE RAJ_ICE;
{-------------------------------------------------------------------------}
{SPECIFICATIONS : Construit et rajoute dans TARGET le paragraphe ICE       }
{       dans la ZONE DES PERFORM                                           }
```

```
{-------------------------------------------------------------------------------}


   BEGIN
     INS('0006A',MESSAGE,TARGET);
   END;



{-------------------------------------------------------------------------------}
 PROCEDURE GET_TYPE_PHRASE;
{-------------------------------------------------------------------------------}
{SPECIFICATIONS :                                                               }
{   Fournit le type d'une phrase                                                }
{-------------------------------------------------------------------------------}

   VAR
     LIGNED: tligne;       {les premiers caractères de la ligne courante}



   BEGIN
     Type_ligne:=aut;
     IF LIGNE=''
          THEN TYPE_LIGNE:=VID;
     LIGNED:=COPY(LIGNE,1,6);
     IF (LIGNED<>MARGE06) and (Type_Ligne<>VID)
          THEN
            BEGIN
              TYPE_LIGNE:=ERR;
              Stop_Erreur('0002E');
            END;

     IF (LIGNED=MARGE06) and (ligne[7]<>'*') AND (ligne[7]<>' ')
          THEN
            BEGIN
              TYPE_LIGNE:=ERR;
              Stop_Erreur('0003E');
            END;

     IF (LIGNED=MARGE06) and (ligne[7]='*')
          THEN TYPE_LIGNE:=COM;

     LIGNED:=COPY(LIGNE,1,11);
     IF LIGNED=MARGE11
          THEN TYPE_LIGNE:=INSTR;

     LIGNED:=COPY(LIGNE,1,9);
     IF (LIGNED=MARGE07+'01')
          THEN TYPE_LIGNE:=D01;

     LIGNED:=COPY(LIGNE,1,7);
     IF (LIGNED=MARGE07) AND (LIGNE[8]<>' ') and (Type_Ligne<>D01)
          THEN {c'est un label ssi le 8eme caractère est une lettre}
```

```
                TYPE_LIGNE:=LAB;

   END;



{-------------------------------------------------------------------------}
 PROCEDURE ICE_ITER;
{-------------------------------------------------------------------------}
{SPECIFICATIONS :                                                         }
{   Traite une ligne du fichier SOURCE                                    }
{-------------------------------------------------------------------------}

   BEGIN
      Get_Type_Phrase;
      RAJ_ENV_DIV;
      RAJ_DATA_DIV;
      RAJ_ICEIPR;
      COMPTE_ECRAN;
      RAJ_ICEITR;
      RAJ_ICEXXX_I;
      RAJ_ICEXXX_T;
      RAJ_ICETPR;
      RAJ_ICETTR;

      IF (Type_ligne<>com) or (not DC0001)
            THEN WRITELN(TARGET,LIGNE);
   END;



{*************************************************************************}
{ COPRS DU PROGRAMME PRINCIPAL                                            }
{*************************************************************************}

 BEGIN
   ICE_INIT;
   CLIGNE:=0;


   WHILE NOT EOF(SOURCE) DO
      BEGIN
         CLIGNE:=CLIGNE+1;
         gotoxy(poscurx,poscury);
         write(cligne);
         READLN(SOURCE,LIGNE);
         Ice_Iter;
      END;
   RAJ_ICEAFF;
   WRITELN;
   writeln;
```

```
   writeln('  Fichier ',upper(NfSour),' précompilé avec succès.');
   writeln('  Génération d''un fichier ',upper(NfTarg));

   ICE_CLOT;

END.

{*****************************************************************************}
{                          FIN DU FICHIER ICE.PAS                             }
{*****************************************************************************}
```

2.3.2 Fichier des messages d'ERREUR du précompilateur

```
|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||||
|NUM=0001E

  ***************************************************************************
  * ERREUR ICE #0001 PARAMETRES MAQUANTS !!!                                *
  * USAGE : ICE <NOM_FICH_SOURCE> [<NOM_FICH_TARGET>] [SANS*]               *
  ***************************************************************************


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||||
|NUM=0002E

  ***************************************************************************
  * ERREUR ICE #0002  !!!                                                   *
  * CARACTERES RENCONTRES AVANT LA 7EME COLONE ...                          *
  ***************************************************************************


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||||
|NUM=0003E

  ***************************************************************************
  * ERREUR ICE #0003  !!!                                                   *
  * CARACTERE DANS LA 7EME COLONE DIFFERENT DE "*" ...                      *
  ***************************************************************************


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||||
|NUM=0004E

  ***************************************************************************
  * ERREUR ICE #0004  !!!                                                   *
  * LABEL INVALIDE ...                                                      *
  ***************************************************************************


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||||
```

```
|NUM=0005E

  ************************************************************************
  * ERREUR ICE #0005  !!!                                                *
  * CARACTERE INVALIDE ...                                               *
  ************************************************************************

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|NUM=0006E

  ************************************************************************
  * ERREUR ICE #0006  !!!                                                *
  * "." PROBABLEMENT MANQUANT ...                                        *
  ************************************************************************

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|NUM=0007E

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|NUM=0008E

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|NUM=0009E

  ************************************************************************
  * ERREUR ICE #0009  !!!                                                *
  * ERREUR SYNTAXIQUE DANS L'ORDRE DES LABELS ...                        *
  ************************************************************************


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|NUM=0010E


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
```

2.3.3 Fichier des MESSAGE du précompilateur

```
|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|POS=ED
|NUM=0002A


      *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
      * ELEMENT DE L'ENVIRONMENT DIVISION RAJOUTER PAR ICE
      *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

       CONFIGURATION SECTION.

           SPECIAL-NAMES.
```

```
           CRT STATUS IS ICECRT.

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
POS=WSS
NUM=0003A



      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
      *   ELEMENT DE LA WORKING-STORAGE SECTION RAJOUTER PAR ICE
      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

       01 ICECRT.
           02 ICECR1 PIC 9  USAGE IS DISPLAY.
           02 ICECR2 PIC 99 COMP.
           02 ICECR3 PIC 99 COMP.
       01 ICECUR PIC 9(03) VALUE   0.
       01 ICENXT PIC 9(03) VALUE   1.
       01 ICEBCH PIC 9(03) VALUE   0.
       01 ICEATR PIC X(01) VALUE 'N'.
       01 ICEEPR PIC X(01) VALUE 'N'.
       01 ICEETR PIC X(01) VALUE 'N'.
       01 ICEFIN PIC X(01) VALUE 'N'.
       01 ICEREF PIC X(01) VALUE 'N'.
       01 ICECCC PIC X(01) VALUE 'I'.
       01 ICEPF1 PIC X(01) VALUE 'N'.

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
POS=DD-SS
NUM=0004A



|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
POS=PD-GTI
NUM=0005A



      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
      * ELEMENT DE LA GENERAL TRANSACTION INITIALISATION RAJOUTER PAR ICE
      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

           PERFORM FBSS-BT.
           MOVE 0 TO ICECUR.
           MOVE 1 TO ICENXT.

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
POS=PD-IOF
NUM=0006A

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
```

```
POS=PD-GTC
NUM=0007A

      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
      * ELEMENT DE LA GENERAL TRANSACTION CLOSURE RAJOUTER PAR ICE
      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

           PERFORM FBSS-ET.
           GO TO ICEITR.

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
POS=PD-GPC
NUM=0008A



      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
      * ELEMENT DE LA GENERAL PROGRAM CLOSURE RAJOUTER PAR ICE
      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

           DISPLAY SPACES AT LINE 1 COL 1.
           DISPLAY "PROGRAMME ABANDONNE SUR DEMANDE DE L'UTILISATEUR".
      *    EXIT PROGRAM.
           STOP RUN.

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
POS=PD-ZP
NUM=0009A

      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
      * ELEMENT DE LA ZONE DES PERFORM RAJOUTER PAR ICE
      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
POS=PD-ZP
NUM=0010A

       ICEAFF.
           MOVE 'N' TO ICEFIN.
           GO TO WICEBOUCLE02.


       WICEBOUCLE02.
           IF ICEFIN = 'Y'
                GO TO ICE
           ELSE
                GO TO WICEAFF.
```

```
 WICEAFF.
*    Affichage et saisie du bon ecran.
     MOVE 'N' TO ICEPF1.
     PERFORM ICEAFFSCR.
*    Traitement des codes de terminaison de la saisie.
     IF ICECR1 = 1 AND ICECR2 = 2
          GO TO ICEPF2.
     IF ICECR1 = 1 AND ICECR2 = 3
          GO TO ICEPF3.

*    Traitement des codes de terminaison de la saisie.
     IF ICEREF = 'Y'
          MOVE 'N' TO ICEFIN
          MOVE 'Y' TO ICEPF1
          PERFORM ICEAFFSCR
     ELSE
          MOVE 'Y' TO ICEFIN.
     IF ICECR1 = 1 AND ICECR2 = 1
          MOVE 'Y' TO ICEFIN.
     IF ICECR1 = 1 AND ICECR2 = 2
          GO TO ICEPF2.
     IF ICECR1 = 1 AND ICECR2 = 3
          GO TO ICEPF3.

     GO TO WICEBOUCLE02.




 ICEPF2.
     MOVE 'N' TO ICEATR.
     PERFORM FBSS-RB.
     MOVE 4 TO ICEBCH.
     GO TO WICEITRD.

 ICEPF3.
     display spaces at line 1 col 1.
     display 'PF3'.
     accept icecr3.
     MOVE 'N' TO ICEEPR.
     PERFORM FBSS-ET.
     MOVE 11 TO ICEBCH.
     GO TO WICETPRD.


 ICE.
     display spaces at line 1 col 1.
     display 'ice ',icebch.
     accept icecr3.
     IF ICEBCH NOT = 7
          COMPUTE ICEBCH = ICEBCH + 1.
     IF ICEEPR = 'Y'
          GO TO ICEPF3.
```

```
    IF ICEATR = 'Y'
          GO TO ICEPF2.
    IF ICEETR = 'Y'
          MOVE 8 TO ICEBCH
          GO TO WICETTRD.

    GO TO WICEIPRD,ICEIPR,WICEIPRF,WICEITRD,ICEITR,WICEITRF,
          WICE000,WICETTRD,ICETTR,WICETTRF,WICETPRD,ICETPR,
          WICETPRF DEPENDING ON ICEBCH.



WICEIPRD.
    PERFORM FBSS-INIT-CPRB-DB.
    PERFORM FBSS-GF.
    PERFORM FBSS-OO.
    GO TO ICE.

WICEIPRF.
    GO TO ICE.

WICEITRD.
    PERFORM FBSS-BT.
    MOVE 0    TO ICECUR.
    MOVE 1    TO ICENXT.
    MOVE 4    TO ICEBCH.
    MOVE 'N'  TO ICEATR.
    MOVE 'N'  TO ICEEPR.
    MOVE 'N'  TO ICEETR.
    MOVE 'N'  TO ICEREF.
    MOVE 'I'  TO ICECCC.
    GO TO ICE.

WICEITRF.
    GO TO ICE.

WICETTRD.
    GO TO ICE.

WICETTRF.
    GO TO ICE.

WICETPRD.
    display spaces at line 1 col 1.
    display 'icetprd'.
    accept icecr3.
    GO TO ICE.



WICETPRF.
    DISPLAY SPACES AT LINE 1 COL 1.
    DISPLAY "PROGRAMME ABANDONNE SUR DEMANDE DE L'UTILISATEUR".
*   EXIT PROGRAM.
```

```
           STOP RUN.


      * INCLUSION DU FICHIER DES ACCES FICHIER

           COPY FBSS-FCT.CBL.

       WICE000.

           IF ICECCC = 'I'
|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|POS=PD-ZP
|NUM=0011A

           GO TO ICE.

|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
|POS=PD-ZP
|NUM=0012A
           GO TO ICEERR01.

       ICEERR01.
           DISPLAY SPACES AT LINE 1 COL 1.
           DISPLAY 'Run time ICE error #001'.
           DISPLAY 'Screen ICE',ICENXT,' does not exist #001'.
           DISPLAY 'Press RETURN to exit program'.
           STOP RUN.


|||||||||||||||||||||||||||||| MESSAGE ICE ||||||||||||||||||||||||||||||||
```

2.4 Une application ICE

2.4.1 AFIC rédigé en ICE

```
*****************************************************************
* Fichier : AFIC.ICE
*
* Contenu : Ce programme permet de réaliser l'appel à la procedure FIC
*           Il doit etre precompiler par ICE-PC.
*
* Auteur : D. Bauchau.
*
* Date : 8 Novembre 89.
*
*****************************************************************

*---------------------------------------------------------------*
 IDENTIFICATION DIVISION.
*---------------------------------------------------------------*
     PROGRAM-ID. AFIC.
     AUTHOR. D-BAUCHAU.

*---------------------------------------------------------------*
 ENVIRONMENT DIVISION.
*---------------------------------------------------------------*


*---------------------------------------------------------------*
 DATA DIVISION.
*---------------------------------------------------------------*

*...............................................................*
 WORKING-STORAGE SECTION.
*...............................................................*

     COPY WDB-INIT.CBL.
     COPY WFICINTR.CBL.

*...............................................................*
 SCREEN SECTION.
*...............................................................*

     COPY SSICE001.CBL.


*---------------------------------------------------------------*
 PROCEDURE DIVISION.
*---------------------------------------------------------------*

*...............................................................*
* GENERAL PROGRAM INITIALISATION
*...............................................................*
 ICEIPR.
```

```
     GO TO ICE.

*.....................................................................*
* GENERAL TRANSACTION INITIALISATION
*.....................................................................*
 ICEITR.
     GO TO ICE.


*.....................................................................*
* INITIALISATION OF FORM1
*.....................................................................*
 ICE001-I.
     GO TO ICE.

*.....................................................................*
* TEST RELATED TO FORM1
*.....................................................................*
 ICE001-T.
     GO TO ICE.

*.....................................................................*
* END OF FORM1
*.....................................................................*
 ICE001-E.
     CALL "FIC" USING FUNCTION, PARAM, DEB, FICH.
     GO TO ICE.
*.....................................................................*
* GENERAL TRANSACTION CLOSURE
*.....................................................................*
 ICETTR.
     GO TO ICE.


*.....................................................................*
* GENERAL PROGRAM CLOSURE
*.....................................................................*
 ICETPR.
     GO TO ICE.

**********************************************************************
*                  FIN DU FICHIER AFIC.ICE                           *
**********************************************************************
**********************************************************************
```

2.4.2 AFIC en COBOL obtenu par précompilation du fichier AFIC en ICE

```
**********************************************************************
* FICHIER REALISE PAR PRECOMPILATION DU FICHIER AFIC.ICE        *
```

```
*****************************************************************
* PRECOMPILE PAR ICE VERSION 0.0 NOVEMBRE 89                    *
*****************************************************************
* LE  8/ 1/1990                                                 *
*****************************************************************
* A  11: 9:47:72                                                *
*****************************************************************


*****************************************************************
* Fichier : AFIC.ICE
*
* Contenu : Ce programme permet de réaliser l'appel à la procedure FIC
*           Il doit etre precompiler par ICE-PC.
*
* Auteur : D. Bauchau.
*
* Date : 8 Novembre 89.
*
*****************************************************************

*---------------------------------------------------------------*
 IDENTIFICATION DIVISION.
*---------------------------------------------------------------*
     PROGRAM-ID. AFIC.
     AUTHOR. D-BAUCHAU.

*---------------------------------------------------------------*
 ENVIRONMENT DIVISION.
*---------------------------------------------------------------*


*---------------------------------------------------------------*


*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
* ELEMENT DE L'ENVIRONMENT DIVISION RAJOUTER PAR ICE
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

 CONFIGURATION SECTION.

     SPECIAL-NAMES.
     CRT STATUS IS ICECRT.

 DATA DIVISION.
*---------------------------------------------------------------*

*...............................................................*
 WORKING-STORAGE SECTION.
*...............................................................*

     COPY WDB-INIT.CBL.
```

```
     COPY WFICINTR.CBL.

*.....................................................................*


*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
*   ELEMENT DE LA WORKING-STORAGE SECTION RAJOUTER PAR ICE
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*

 01 ICECRT.
     02 ICECR1 PIC 9  USAGE IS DISPLAY.
     02 ICECR2 PIC 99 COMP.
     02 ICECR3 PIC 99 COMP.
 01 ICECUR PIC 9(03) VALUE   0.
 01 ICENXT PIC 9(03) VALUE   1.
 01 ICEBCH PIC 9(03) VALUE   0.
 01 ICEATR PIC X(01) VALUE 'N'.
 01 ICEEPR PIC X(01) VALUE 'N'.
 01 ICEETR PIC X(01) VALUE 'N'.
 01 ICEFIN PIC X(01) VALUE 'N'.
 01 ICEREF PIC X(01) VALUE 'N'.
 01 ICECCC PIC X(01) VALUE 'I'.
 01 ICEPF1 PIC X(01) VALUE 'N'.

 SCREEN SECTION.
*.....................................................................*

     COPY SSICE001.CBL.


*---------------------------------------------------------------------*
 PROCEDURE DIVISION.
*---------------------------------------------------------------------*

*.....................................................................*
* GENERAL PROGRAM INITIALISATION
*.....................................................................*

     GO TO ICE.

 ICEIPR.
     GO TO ICE.


*.....................................................................*
* GENERAL TRANSACTION INITIALISATION
*.....................................................................*
 ICEITR.
     GO TO ICE.


*.....................................................................*
```

```
* INITIALISATION OF FORM1
*.....................................................................*
 ICE001-I.
     GO TO ICE.

*.....................................................................*
* TEST RELATED TO FORM1
*.....................................................................*
 ICE001-T.
     GO TO ICE.

*.....................................................................*
* END OF FORM1
*.....................................................................*
 ICE001-E.
     CALL "FIC" USING FUNCTION, PARAM, DEB, FICH.
     GO TO ICE.
*.....................................................................*
* GENERAL TRANSACTION CLOSURE
*.....................................................................*
 ICETTR.
     GO TO ICE.


*.....................................................................*
* GENERAL PROGRAM CLOSURE
*.....................................................................*
 ICETPR.
     GO TO ICE.

*.....................................................................*
* ZONE DES PERFORM
*.....................................................................*

**********************************************************************
*                  FIN DU FICHIER AFIC.ICE                           *
**********************************************************************
**********************************************************************




*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
* ELEMENT DE LA ZONE DES PERFORM RAJOUTER PAR ICE
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*


 ICEAFFSCR.
    IF ICECUR = 001
         DISPLAY ICE001
         IF ICEPF1 = 'Y'
              DISPLAY 'PF1 pour confirmer'
              AT LINE 24 COL 1
```

```
            WITH REVERSE-VIDEO.
         ACCEPT  ICE001.

 ICEAFF.
     MOVE 'N' TO ICEFIN.
     GO TO WICEBOUCLE02.


 WICEBOUCLE02.
     IF ICEFIN = 'Y'
          GO TO ICE
     ELSE
          GO TO WICEAFF.



 WICEAFF.
*    Affichage et saisie du bon ecran.
     MOVE 'N' TO ICEPF1.
     PERFORM ICEAFFSCR.
*    Traitement des codes de terminaison de la saisie.
     IF ICECR1 = 1 AND ICECR2 = 2
           GO TO ICEPF2.
     IF ICECR1 = 1 AND ICECR2 = 3
           GO TO ICEPF3.

*    Traitement des codes de terminaison de la saisie.
     IF ICEREF = 'Y'
          MOVE 'N' TO ICEFIN
          MOVE 'Y' TO ICEPF1
          PERFORM ICEAFFSCR
     ELSE
          MOVE 'Y' TO ICEFIN.
     IF ICECR1 = 1 AND ICECR2 = 1
           MOVE 'Y' TO ICEFIN.
     IF ICECR1 = 1 AND ICECR2 = 2
           GO TO ICEPF2.
     IF ICECR1 = 1 AND ICECR2 = 3
           GO TO ICEPF3.

     GO TO WICEBOUCLE02.




 ICEPF2.
     MOVE 'N' TO ICEATR.
     PERFORM FBSS-RB.
     MOVE 4 TO ICEBCH.
     GO TO WICEITRD.

 ICEPF3.
     display spaces at line 1 col 1.
```

```
    display 'PF3'.
    accept icecr3.
    MOVE 'N' TO ICEEPR.
    PERFORM FBSS-ET.
    MOVE 11 TO ICEBCH.
    GO TO WICETPRD.


ICE.
    display spaces at line 1 col 1.
    display 'ice ',icebch.
    accept icecr3.
    IF ICEBCH NOT = 7
          COMPUTE ICEBCH = ICEBCH + 1.
    IF ICEEPR = 'Y'
          GO TO ICEPF3.
    IF ICEATR = 'Y'
          GO TO ICEPF2.
    IF ICEETR = 'Y'
          MOVE 8 TO ICEBCH
          GO TO WICETTRD.

    GO TO WICEIPRD,ICEIPR,WICEIPRF,WICEITRD,ICEITR,WICEITRF,
          WICE000,WICETTRD,ICETTR,WICETTRF,WICETPRD,ICETPR,
          WICETPRF DEPENDING ON ICEBCH.



WICEIPRD.
    PERFORM FBSS-INIT-CPRB-DB.
    PERFORM FBSS-GF.
    PERFORM FBSS-OO.
    GO TO ICE.

WICEIPRF.
    GO TO ICE.

WICEITRD.
    PERFORM FBSS-BT.
    MOVE 0    TO ICECUR.
    MOVE 1    TO ICENXT.
    MOVE 4    TO ICEBCH.
    MOVE 'N'  TO ICEATR.
    MOVE 'N'  TO ICEEPR.
    MOVE 'N'  TO ICEETR.
    MOVE 'N'  TO ICEREF.
    MOVE 'I'  TO ICECCC.
    GO TO ICE.

WICEITRF.
    GO TO ICE.

WICETTRD.
    GO TO ICE.
```

```
WICETTRF.
    GO TO ICE.

WICETPRD.
    display spaces at line 1 col 1.
    display 'icetprd'.
    accept icecr3.
    GO TO ICE.



WICETPRF.
    DISPLAY SPACES AT LINE 1 COL 1.
    DISPLAY "PROGRAMME ABANDONNE SUR DEMANDE DE L'UTILISATEUR".
*   EXIT PROGRAM.
    STOP RUN.


* INCLUSION DU FICHIER DES ACCES FICHIER

    COPY FBSS-FCT.CBL.

WICE000.

    IF ICECCC = 'I'
             GO TO WICE001 DEPENDING ON ICENXT
    ELSE     GO TO WICE001 DEPENDING ON ICECUR.
    GO TO ICEERR01.

ICEERR01.
    DISPLAY SPACES AT LINE 1 COL 1.
    DISPLAY 'Run time ICE error #001'.
    DISPLAY 'Screen ICE',ICENXT,' does not exist #001'.
    DISPLAY 'Press RETURN to exit program'.
    STOP RUN.


WICE001.
    IF ICECCC = 'I'
             MOVE 001 TO ICECUR
             MOVE 'T' TO ICECCC
             MOVE 'N' TO ICEREF
             GO TO ICE001-I.
    IF ICECCC = 'T'
             MOVE 001 TO ICECUR
             MOVE 'D' TO ICECCC
             GO TO ICEAFF.
    IF ICECCC = 'D'
             MOVE 001 TO ICECUR
             MOVE 'E' TO ICECCC
             GO TO ICE001-T.
    IF ICECCC = 'E'
             MOVE 001 TO ICECUR
```

```
                MOVE 'F' TO ICECCC
                GO TO ICE001-E.
        IF ICECCC = 'F'
                MOVE 001 TO ICECUR
                MOVE 'I' TO ICECCC.
        GO TO ICE.
```

2.4.3 AFIC en COBOL standard

```
*****************************************************************
* Fichier : AFIC.CBL
*
* Contenu : Ce programme permet de réaliser l'appel à la procedure FIC
*
* Auteur : D. Bauchau.
*
* Date : 6 octobre 89.
*
*****************************************************************


*---------------------------------------------------------------*
 IDENTIFICATION DIVISION.
*---------------------------------------------------------------*
   PROGRAM-ID. AFIC.
   AUTHOR. D-BAUCHAU.



*---------------------------------------------------------------*
 DATA DIVISION.
*---------------------------------------------------------------*

*...............................................................*
 WORKING-STORAGE SECTION.
*...............................................................*

 COPY WSVPSAIS.CBL.
 COPY WDB-INIT.CBL.
 COPY WFICINTR.CBL.

 01 fin       pic x(4).

*...............................................................*
 screen section.
*...............................................................*


* COPY SSVPSAIS.CBL.
* COPY SSVPAFFR.CBL.

 01 ecran.
```

```
       02 BLANK SCREEN.

       02 LINE 1    COL 1  HIGHLIGHT
                           VALUE "Test Procedure FIC".

       02 LINE + 1  COL 1  VALUE "Function id              :".
       02           COL 30 REVERSE-VIDEO
                           PIC X(2)  USING function AUTO.

       02 LINE + 1  COL 1  value "Parametre (LOck/NoLock) : ".
       02           COL 30 REVERSE-VIDEO
                           PIC X(2)  USING PARAM AUTO.


       02 LINE + 1  COL 1  value "Nom du fichier           : ".
       02           COL 30 REVERSE-VIDEO
                           PIC X(6)  USING NOMFIC AUTO.

       02 LINE + 1  COL 1  value "Numero de la clef        : ".
       02           COL 30 REVERSE-VIDEO
                           PIC X(1) USING NUMCLE AUTO.

       02 LINE + 1  COL 1  value "Enregistrement           :  ".
       02           COL 30 REVERSE-VIDEO
                           PIC X(1024) USING enreg AUTO.

       02 LINE + 1  COL 1  value "Code erreur              :  ".
       02           COL 30 REVERSE-VIDEO
                           PIC 9(2) USING RETCOD AUTO.

       02 LINE + 1  COL 1  value "Code erreur sup          :  ".
       02           COL 30 REVERSE-VIDEO
                           PIC X(2) USING DEB AUTO.


*-----------------------------------------------------------------*
 PROCEDURE DIVISION.
*-----------------------------------------------------------------*

 DEBUT-AFIC.

   PERFORM FBSS-INIT-CPRB-DB.
   PERFORM FBSS-GF.
   PERFORM FBSS-OO.
   MOVE "FAUX" TO FIN.

   PERFORM BOUCLE UNTIL FIN= "VRAI".

 STOP RUN.

 COPY FBSS-FCT.CBL.

 BOUCLE.
   display ecran.
```

```
   accept  ecran ON ESCAPE MOVE "VRAI" TO FIN.
   CALL "FIC" USING FUNCTION, PARAM, DEB, FICH.

*****************************************************************
*                      FIN DU FICHIER AFIC.CBL                  *
*****************************************************************
*****************************************************************
```

2.4.4 SCREEN SECTION de l'écran 001

```
01 ICE001.

   02 BLANK SCREEN.

   02 LINE 1     COL 1  HIGHLIGHT
                     VALUE "Test Procedure FIC".

   02 LINE + 1   COL 1  VALUE "Function id             :".
   02            COL 30 REVERSE-VIDEO
                     PIC X(2)  USING function AUTO.

   02 LINE + 1   COL 1  value "Parametre (LOck/NoLock) : ".
   02            COL 30 REVERSE-VIDEO
                     PIC X(2)  USING PARAM AUTO.


   02 LINE + 1   COL 1  value "Nom du fichier          : ".
   02            COL 30 REVERSE-VIDEO
                     PIC X(6)  USING NOMFIC AUTO.

   02 LINE + 1   COL 1  value "Numero de la clef       : ".
   02            COL 30 REVERSE-VIDEO
                     PIC X(1) USING NUMCLE AUTO.

   02 LINE + 1   COL 1  value "Enregistrement          :  ".
   02            COL 30 REVERSE-VIDEO
                     PIC X(1024) USING enreg AUTO.

   02 LINE + 1   COL 1  value "Code erreur             :  ".
   02            COL 30 REVERSE-VIDEO
                     PIC 9(2) USING RETCOD AUTO.

   02 LINE + 1   COL 1  value "Code erreur sup         :  ".
   02            COL 30 REVERSE-VIDEO
                     PIC X(2) USING DEB AUTO.
```

3 Une application en ICE-DBASE

PC____.PRG

```
* ======================
* IDENTIFICATION DIVISION.
* ======================

*   Nom pgm : PC....
*   Auteur  : Hofmans Pierre
*   Dates   : version 1.0    10/05/90

*   Utilite du pgm :

*   Fichiers utilises :
*   """""""""""""""""""
*      - en INPUT  : SELECT 1  -  PCXD10  -  Messages d'erreur
*                    SELECT 2  -  PCXD20  -  Dictionnaire
*                    SELECT 3  -  PCXD94  -  Repertoire des fichiers
*      - en OUTPUT : SELECT 4  -  PCXLOG  -  Fichier "logfile"
*      - en I-O    :


* ====================
* ENVIRONMENT DIVISION.
* ====================
  Clear all
  Set Procedure To PCXPRO
  Do PCXSET


* =============
* DATA DIVISION.
* =============

* Description et initialisation des zones de travail


* Initialisation des zones systèmes et ouverture des fichiers standards
  Do PCXINI


* ==================
* PROCEDURE DIVISION.
* ==================

* PC-IPR.
* ======
* Debut du programme : ouverture des fichiers


* PC-ITR.
* ======
* Debut de la transaction logique

  Do While pcxitr

     pcxttr = .T.
     pcxsmb = 1

* Initialisation des zones de la transaction


* Execution des maps

     Do While pcxitr .And. pcxttr

        Do Case

           Case pcxnex = "A"  → écran simple
                Do PC....AI
                Do PC....AT
```

```
            Do PC....AE

         Case pcxnex = "B"    -> écran sommaire
              Do PC....BI
              Do PC....BT
              Do PC....BE

         Case pcxnex = "C"
              Begin Transaction
              Do PC....CI
              Do PC....CT
              Do PC....CE

         Case pcxnex = "D"
              Do PC....DI
              Do PC....DT
              Do PC....DE

         Case pcxnex = "0"    -> fin normale de la transaction
              maj des fichiers
              End Transaction
              pcxttr = .F.

         Case pcxnex = "2"    -> en cas de PF2 -> retour à l'écran A
              End Transaction    via le $ PC-ITR
              pcxttr = .F.

         Case pcxnex = "3"    -> en cas de PF3 = abandon du pgm.
              End Transaction
              Return          <- Do PCXUST
                                 CLOSE ALL
         Otherwise
              pcxttr = .F.

       Endcase

     Enddo

* Si l'utilisateur a tape PF2 --> retour au debut de la transaction

    If pcxnex = "2"
       pcxnex = "A"
       pcxsmb = 1
       Loop
    Endif

    modification de pcxitr, pcxnex et pcxsmb en fonction de ...

  Enddo


* PC-EPR.
* ======
* Fin du programme

  Do PCXUST
  Close All

  Return


* ------------------------------ Debut MAP A -------------------------------

* ==================
  PROCEDURE PC....AI
* ==================

  initialisation des zones de la map A du pgm PC....
```

```
   Return


 * ==================
   PROCEDURE PC....AT
 * ==================

 * Initialisation des touches fonctions autorisees

   On Key Label F2   Do PCXPF2
   On Key Label F3   Do PCXPF3


 * Boucle : traitement de l'ecran :

   Do While .T.

 * .. Affichage / reception de l'ecran
      Set Format To PC....A
      Close Format

 * .. Verifie si l'utilisateur a tape PF2 ou PF3
      If pcxnex $ "23"
         Exit
      Endif

 * .. Verifie la validite des zones-ecran
      verif. validite ecran :
          si ko : pcxmes = "...."
                  pcxme2 = "x(20)"
                  pcxlog = " "
                  Do PCXMES
                  If pcxsev = "E"
                     Loop
                  Endif

      Exit

   Enddo


 * Suppression des touches fonctions autorisees

   On Key Label F2
   On Key Label F3

   Return


 * ==================
   PROCEDURE PC....AE
 * ==================

   If .Not. pcxnex $ "23"
      init. zones "cle-start" de la map B  si fonction de la map A.
      pcxnex = "B"
   Endif

   Return

 * ------------------------------ Fin MAP A ---------------------------------


 * ------------------------------ Debut MAP B ---------------------------------

 * ==================
   PROCEDURE PC....BI
 * ==================
```

```
   initialisation des zones de la map B du pgm PC....

   Return


  * ==================
    PROCEDURE PC....BT
  * ==================

  * Initialisation des touches fonctions autorisees

    On Key Label F2      Do PCXPF2
    On Key Label F3      Do PCXPF3
    On Key Label F7      Do PCXPF7
    On Key Label F8      Do PCXPF8


  * Boucle : traitement de l'ecran :

    Do While .T.

       pcxpfk = 0

  * .. Affichage / reception de l'ecran
       Set Format To PC....B    -> en principe affichage d'une seule zone : " votre choix : "_
       Close Format

  * .. Verifie si l'utilisateur a tape PF2 ou PF3
       If pcxnex $ "23"
          Exit
       Endif

  * .. Verifie la validite de la touche PF7 et execute le traitement adapte
       If pcxpfk = 7
          If pcxsmb = 1
             pcxmes = "...."
             pcxme2 = space(20)
             pcxlog = " "
             Do PCXMES
             If pcxsev = "E"
                Loop
             Endif
          Else
             pcxsmb = pcxsmb - 1
             Skip -18 (si 18 lignes / écran)
             init. zones "cle-start" de la map B
             Exit
          Endif
       Endif

  * .. Verifie la validite de la touche PF8 et execute le traitement adapte
       If pcxpfk = 8
          If derniere ligne ecran = " "  <=> fin de fichier atteinte
             pcxmes = "...."
             pcxme2 = space(20)
             pcxlog = " "
             Do PCXMES
             If pcxsev = "E"
                Loop
             Endif
          Else
             pcxsmb = pcxsmb + 1
             init. zones "cle-start" de la map B avec derniere ligne ecran
             Exit
          Endif
       Endif

  * .. Verifie la validite des zones-ecran
       verif. validite ecran :
```

```
          si ko : pcxmes = "...."
                  pcxme2 = "x(20)"
                  pcxlog = " "
                  Do PCXMES
                  If pcxsev = "E"
                     Loop
                  Endif

      Exit

   Enddo


 * Suppression des touches fonctions autorisees

   On Key Label F2
   On Key Label F3
   On Key Label F7
   On Key Label F8

   Return


 * ==================
   PROCEDURE PC....BE
   ==================

   If .Not. pcxnex $ "23" .And. .Not. pcxpfk $ "[illegible]"
      init. zones "cle-start" de la map C
      pcxnex = "C"
   Endif

   Return

 * ------------------------------- Fin MAP B ---------------------------------


 * ------------------------------- Debut MAP C ---------------------------------

 * ==================
   PROCEDURE PC....CI
 * ==================

   initialisation des zones de la map C du pgm PC....

   Return

 * ==================
   PROCEDURE PC....CT
 * ==================

 * Initialisation des touches fonctions autorisees

   On Key Label F2   Do PCXPF2
   On Key Label F3   Do PCXPF3


 * Boucle : traitement de l'ecran :

   Do While .T.
      pcxpfk = 0

 * .. Affichage / reception de l'ecran
      Set Format To PC....C
      Close Format

 * .. Verifie si l'utilisateur a tape PF2 ou PF3
      If pcxnex $ "23"
         Exit
```

```
      Endif

*  .. Verifie la validite des zones-ecran
      verif. validite ecran :
          si ko : pcxmes = "...."
                  pcxme2 = "x(20)"
                  pcxlog = " "
                  Do PCXMES
                  If pcxsev = "E"
                     Loop
                  Endif

*  .. Demande de confirmation via la touche PF1
      On Key Label F1   Do PCXPF1

      pcxmes = "...."
      pcxme2 = space(20)
      pcxlog = " "
      Do PCXMES

      On Key Label F1

      If pcxpfk <> 1
         Loop
      Endif

      Exit

   Enddo


*  Suppression des touches fonctions autorisees

   On Key Label F2
   On Key Label F3

   Return


*  ==================
   PROCEDURE PC....CE
*  ==================

   If .Not. pcxnex $ "23"
      init. zones "cle-start" de la map D
      pcxnex = "D"
   Endif

   Return

*  ------------------------------- Fin MAP C ---------------------------------


*  ------------------------------- Debut MAP D -------------------------------

*  ==================
   PROCEDURE PC....DI
*  ==================

   initialisation des zones de la map D du pgm PC....

   Return


*  ==================
   PROCEDURE PC....DT
*  ==================
```

```
* Initialisation des touches fonctions autorisees

  On Key Label F2   Do PCXPF2
  On Key Label F3   Do PCXPF3


* Boucle : traitement de l'ecran :

  Do While .T.

     pcxpfk = 0

* .. Affichage / reception de l'ecran
     Set Format To PC....D
     Close Format

* .. Verifie si l'utilisateur a tape PF2 ou PF3
     If pcxnex $ "23"
        Exit
     Endif

* .. Verifie la validite des zones ecran
     verif. validite ecran :
         si ko : pcxmes = "...."
                 pcxme2 = "x(20)"
                 pcxlog = " "
                 Do PCXMES
                 If pcxsev = "E"
                    Loop
                 Endif


* .. Demande de confirmation via la touche PF1
     On Key Label F1   Do PCXPF1

     pcxmes = "...."
     pcxme2 = space(20)
     pcxlog = " "
     Do PCXMES

     On Key Label F1

     If pcxpfk <> 1
        Loop
     Endif

     Exit

  Enddo


* Suppression des touches fonctions autorisees

  On Key Label F2
  On Key Label F3

  Return


* ==================
  PROCEDURE PC....DE
* ==================

  If .Not. pcxnex $ "23"
     pcxnex = "0"
  Endif

  Return
```

4 Le résultat de la précompilation du programme ICE (Fig III 4)

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. Z20M.
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                                                               *
      *   TITULAIRES TELEPHONIE : DETAIL                              *
      *   ------------------------------                              *
      *       - MAP 1 : DETAIL                                        *
      *                                                               *
      *   PGM APPELE PAR Z20S EN TASK-SWITCH                          *
      *                                                               *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
       AUTHOR. PJO.
      *
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      *
      *    ZONES A SAUVER
      *    --------------
       01  ICEREC.
           03  WORKSV.
               05  LISCOM      PIC X.
               05  LISCAD      PIC X.
               05  LISSER      PIC X.
               05  EXTENS.
                   07  EXTEN1      PIC X VALUE SPACE.
                   07  EXTEN3      PIC 9(3).
      *
      *    DEFINITION DES ECRANS
      *    - - - - - - - - - - -
           COPY Z20M01   SUPPRESS REPLACING Z20M01I BY ICE001
                                            Z20M01O BY ICE101
                                            Z20M01L BY ICE201
                                            Z20M01M BY ICE301.

      *
      *    DEFINITION DES FICHIERS
      *    - - - - - - - - - - - - -
           COPY BZZD20.
             04  FILLER                  PIC  X(094).

      *
      *    INFORMATIONS ENVOYEES ENTRE MODULES
      *    - - - - - - - - - - - - - - - - - -
      *        MODULE Z20M          --> MODULE Z20M
      *    ACTION + CLE

           03  SAVZON.
               05 SAVACT   PIC  X.
                  88  CONSULT VALUE 'C'.
                  88  MODIF   VALUE 'M'.
                  88  AJOUT   VALUE 'A'.
                  88  SUPPR   VALUE 'S'.
               05 SAVNUM   PIC X.
                  88  CLE02   VALUE '2'.
```

```
                88  CLE03    VALUE '3'.
                88  CLE04    VALUE '4'.
             05 SAVCLE       PIC X(10).
             05 SAVCLE22     REDEFINES SAVCLE.
                07 SAVCLE2      PIC X(4).
                07 FILLER       PIC X(6).
             05 SAVCLE33     REDEFINES SAVCLE.
                07 SAVCLE3      PIC X(5).
                07 FILLER       PIC X(5).
             05 SAVCLE44     REDEFINES SAVCLE.
                07 SAVCLE4      PIC X(10).
             05 EXTENS.
                07 EXTEN1       PIC X.
                07 EXTEN3       PIC X(3).
             05 MATRIC       PIC X(5).

      01  ICEREC-END PIC X.
      01  WORKER.
          03  RECPER      PIC 9(3).
          COPY BZZD01.
            04  FILLER                PIC  X(094).

      /
       01  ICEMAP.
           02  FILLER  PIC X(7) VALUE 'Z20M01 '.
       01  ICEMAPZ REDEFINES ICEMAP.
           02  MAPICE  PIC X(7) OCCURS   1.
           COPY ICEITF    SUPPRESS.
           COPY ICECONST SUPPRESS.
           COPY ICESBR    SUPPRESS.
           COPY ICEDSA    SUPPRESS.
       LINKAGE SECTION.
           COPY ICETCTUA SUPPRESS.
      /
       PROCEDURE DIVISION.
           GO TO ICE-START.

       ICEIPR.
           MOVE ICEZON TO SAVZON.
           IF CONSULT
      *       MOVE 'CONSULTATION DES TITULAIRES' TO ICELPG<RLC,30,'C'>
              MOVE SPACES TO ICESBR
              MOVE 'OD' TO ICE-CASBRCOD
              MOVE 'CONSULTATION DES TITULAIRES' TO ICE-CAODDAT
              MOVE 30 TO ICE-CAODOP1
              MOVE 'C' TO ICE-CAODOP2
              MOVE SPACES TO ICE-CAODOP3
              MOVE 'ICESBRB' TO ICENAM
              MOVE +174 TO ICE-1
              CALL 'ICELNK' USING ICENAM
                                  ICESBR
                                  ICE-1
                                  ICECWAA
                                  DFHEIBLK
                                  ICEDSR
```

```
                MOVE ICE-CATO-80 TO ICELPG
             ELSE
           *    MOVE 'MISE-A-JOUR DES TITULAIRES'  TO ICELPG<RLC,30,'C'>.
                MOVE SPACES TO ICESBR
                MOVE 'OD' TO ICE-CASBRCOD
                MOVE 'MISE-A-JOUR DES TITULAIRES' TO ICE-CAODDAT
                MOVE 30 TO ICE-CAODOP1
                MOVE 'C' TO ICE-CAODOP2
                MOVE SPACES TO ICE-CAODOP3
                MOVE 'ICESBRB' TO ICENAM
                MOVE +174 TO ICE-1
                CALL 'ICELNK' USING ICENAM
                                    ICESBR
                                    ICE-1
                                    ICECWAA
                                    DFHEIBLK
                                    ICEDSR
                MOVE ICE-CATO-80 TO ICELPG.
             GO TO ICE.

           *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
           *                    DEBUT DE TRANSACTION                         *
           *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
            ICEITR.
                MOVE EXTENS OF SAVZON TO EXTENS OF CLEFIC OF BZZD20.
                MOVE MATRIC OF SAVZON TO MATRIC OF CLEFIC OF BZZD20.
                IF MODIF
           *       CALL 'ICEFIC' USING ZREADIND BZZD20 ZLOCK
                   MOVE ZREADIND    TO ICEFCT
                   MOVE BZZD20      TO ICEFILEZON
                   MOVE ZLOCK       TO ICETYP
                   ADD  110         TO LONENR-SAVE
                   MOVE 'ICEDATAB' TO ICENAM
                   MOVE LONENR-SAVE TO ICE-1
                   CALL 'ICELNK' USING ICENAM
                                       BZZD20
                                       ICE-1
                                       ICECWAA
                                       DFHEIBLK
                                       ICEDSD
                                       ICETCTUA
                   PERFORM ICE-TEST
                ELSE
           *       CALL 'ICEFIC' USING ZREADIND BZZD20.
                   MOVE ZREADIND    TO ICEFCT
                   MOVE BZZD20      TO ICEFILEZON
                   MOVE ZLIVE       TO ICETYP
                   ADD  110         TO LONENR-SAVE
                   MOVE 'ICEDATAB' TO ICENAM
                   MOVE LONENR-SAVE TO ICE-1
                   CALL 'ICELNK' USING ICENAM
                                       BZZD20
                                       ICE-1
                                       ICECWAA
                                       DFHEIBLK
```

```
                                        ICEDSD
                                        ICETCTUA
              PERFORM ICE-TEST.
           IF RETCOD OF BZZD20 = 12
              MOVE 'XXXX0043' TO ICEMES
      *       ENREGISTREMENT BLOQUE
              PERFORM ICE.
           IF RETCOD OF BZZD20 = 0 AND AJOUT
              MOVE 'XXXX0021' TO ICEMES
      *       ENREGISTREMENT EXISTANT
              PERFORM ICE.
           IF RETCOD OF BZZD20 NOT = 0 AND NOT AJOUT
              MOVE 'XXXX0045' TO ICEMES
      *       ENREGISTREMENT INEXISTANT
              PERFORM ICE.

           GO TO ICE.

      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                    M A P  1                                     *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
       ICE001-I.
           MOVE LOW-VALUES TO ICE101.
           PERFORM ICEMCO-101-301.
           MOVE SPACES                TO ICE001.
      *    MOVE ICECRS TO NOMALPI OF ICE001.
           CALL 'ICEMCODC' USING ICE201 ICE301 ICE301
           MULTIPLY -1 BY NOMALPL OF ICE201
           CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301.
           IF AJOUT
              MOVE 'AJOUT' TO ICEHD3
              MOVE SPACE TO ENREGI OF BZZD20.
           IF CONSULT
              MOVE 'CONSULTATION' TO ICEHD3
              PERFORM PRO-SCR1.
           IF MODIF
              MOVE 'MISE-A-JOUR' TO ICEHD3.
           PERFORM FILL-SCR1.
           GO TO ICE.

       ICE001-T.
           CALL 'ICESPC' USING ICE001 ICE101.
           IF CONSULT
              GO TO ICE.
           MOVE 'Y' TO ICEREF.

           IF SUPPREI OF ICE001 NOT = '0' AND NOT = 'N'
      *       MOVE ICEREQ TO SUPPREI OF ICE001
              MOVE ICERED TO SUPPREC OF ICE301
              MOVE ICEREV TO SUPPREH OF ICE301
              CALL 'ICEMCOD3' USING ICE201 ICE301 ICE301
              MOVE ICECRS TO SUPPREL OF ICE301
              MOVE 'ZXXX0001' TO ICEMES
      *       DOIT ETRE O OU N
      *       PERFORM ICE.
```

10

```
           PERFORM ICE
           MOVE SUPPREC OF ICE101
             TO SUPPREC OF ICE301
           MOVE SUPPREH OF ICE101
             TO SUPPREH OF ICE301
           CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301.
         IF SUPPREI OF ICE001 = '0'
           MOVE 'S' TO SAVACT
           MOVE 'SUPPRESSION' TO ICEHD3
           MOVE EXTEN3 OF SAVZON TO EXTENSI OF ICE001
           GO TO ICE.

      *  MOVE EXTENSI OF ICE001 <NUM,3>  TO EXTEN3 OF WORKSV.
         MOVE SPACES TO ICESBR
         MOVE 'I2' TO ICE-CASBRCOD
         MOVE EXTENSI OF ICE001 TO ICE-CAI2DAT
         MOVE 3 TO ICE-CAI2OP1
         MOVE 'ICESBRB' TO ICENAM
         MOVE +174 TO ICE-1
         CALL 'ICELNK' USING ICENAM
                             ICESBR
                             ICE-1
                             ICECWAA
                             DFHEIBLK
                             ICEDSR
         MOVE ICERED TO EXTENSC OF ICE301
         MOVE ICEREV TO EXTENSH OF ICE301
         CALL 'ICEMCOD3' USING ICE201 ICE301 ICE301
         MOVE ICECRS TO EXTENSL OF ICE301
         PERFORM ICESBR-TEST
         MOVE EXTENSC OF ICE101
           TO EXTENSC OF ICE301
         MOVE EXTENSH OF ICE101
           TO EXTENSH OF ICE301
         CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301
         MOVE ICE-CATO-17 TO EXTEN3 OF WORKSV.
         MOVE EXTEN3  OF WORKSV          TO EXTENSI OF ICE001.

         MOVE '01' TO NUMTAB OF BZZD01.
         MOVE SERVICI OF ICE001          TO CLETAB OF BZZD01.
      *  CALL 'ICEFIC' USING ZREADIND BZZD01.
         MOVE ZREADIND   TO ICEFCT
         MOVE BZZD01     TO ICEFILEZON
         MOVE ZLIVE      TO ICETYP
         ADD  110        TO LONENR-SAVE
         MOVE 'ICEDATAB' TO ICENAM
         MOVE LONENR-SAVE TO ICE-1
         CALL 'ICELNK' USING ICENAM
                             BZZD01
                             ICE-1
                             ICECWAA
                             DFHEIBLK
                             ICEDSD
                             ICETCTUA
         PERFORM ICE-TEST.
```

11

```
           IF RETCOD OF BZZD01  = 0
              MOVE ABGSER OF BZZD01 TO ABGSERI OF ICE001
           ELSE
              MOVE SPACES TO ABGSERI OF ICE001
      *       MOVE ICEREQ TO SERVICI OF ICE001
              MOVE ICERED TO SERVICC OF ICE301
              MOVE ICEREV TO SERVICH OF ICE301
              CALL 'ICEMCOD3' USING ICE201 ICE301 ICE301
              MOVE ICECRS TO SERVICL OF ICE301
              MOVE 'ZXXX0002' TO ICEMES
      *       SERVICE INEXISTANT
      *       PERFORM ICE.
              PERFORM ICE
              MOVE SERVICC OF ICE101
                TO SERVICC OF ICE301
              MOVE SERVICH OF ICE101
                TO SERVICH OF ICE301
              CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301.

           IF RECPERI OF ICE001 NOT = SPACES
      *       MOVE RECPERI OF ICE001 <NUM,3> TO RECPER OF WORKER
              MOVE SPACES TO ICESBR
              MOVE 'I2' TO ICE-CASBRCOD
              MOVE RECPERI OF ICE001 TO ICE-CAI2DAT
              MOVE 3 TO ICE-CAI2OP1
              MOVE 'ICESBRB' TO ICENAM
              MOVE +174 TO ICE-1
              CALL 'ICELNK' USING ICENAM
                                  ICESBR
                                  ICE-1
                                  ICECWAA
                                  DFHEIBLK
                                  ICEDSR
              MOVE ICERED TO RECPERC OF ICE301
              MOVE ICEREV TO RECPERH OF ICE301
              CALL 'ICEMCOD3' USING ICE201 ICE301 ICE301
              MOVE ICECRS TO RECPERL OF ICE301
              PERFORM ICESBR-TEST
              MOVE RECPERC OF ICE101
                TO RECPERC OF ICE301
              MOVE RECPERH OF ICE101
                TO RECPERH OF ICE301
              CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301
              MOVE ICE-CATO-17 TO RECPER OF WORKER
              MOVE RECPER OF WORKER TO RECPERI OF ICE001.

      *    MOVE LISCOMI OF ICE001 <SWI> TO LISCOM OF WORKSV.
           MOVE SPACES TO ICESBR
           MOVE 'I1' TO ICE-CASBRCOD
           MOVE LISCOMI OF ICE001 TO ICE-CAI1DAT
           MOVE 'D' TO ICE-CAI1OP1
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
```

```
                                ICE-1
                                ICECWAA
                                DFHEIBLK
                                ICEDSR
           MOVE ICERED TO LISCOMC OF ICE301
           MOVE ICEREV TO LISCOMH OF ICE301
           CALL 'ICEMCOD3' USING ICE201 ICE301 ICE301
           MOVE ICECRS TO LISCOML OF ICE301
           PERFORM ICESBR-TEST
           MOVE LISCOMC OF ICE101
             TO LISCOMC OF ICE301
           MOVE LISCOMH OF ICE101
             TO LISCOMH OF ICE301
           CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301
           MOVE ICE-CATO-1  TO LISCOM OF WORKSV.
       *   MOVE LISCADI OF ICE001 <SWI> TO LISCAD OF WORKSV.
           MOVE SPACES TO ICESBR
           MOVE 'I1' TO ICE-CASBRCOD
           MOVE LISCADI OF ICE001 TO ICE-CAI1DAT
           MOVE 'D' TO ICE-CAI1OP1
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICERED TO LISCADC OF ICE301
           MOVE ICEREV TO LISCADH OF ICE301
           CALL 'ICEMCOD3' USING ICE201 ICE301 ICE301
           MOVE ICECRS TO LISCADL OF ICE301
           PERFORM ICESBR-TEST
           MOVE LISCADC OF ICE101
             TO LISCADC OF ICE301
           MOVE LISCADH OF ICE101
             TO LISCADH OF ICE301
           CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301
           MOVE ICE-CATO-1  TO LISCAD OF WORKSV.
       *   MOVE LISSERI OF ICE001 <SWI> TO LISSER OF WORKSV.
           MOVE SPACES TO ICESBR
           MOVE 'I1' TO ICE-CASBRCOD
           MOVE LISSERI OF ICE001 TO ICE-CAI1DAT
           MOVE 'D' TO ICE-CAI1OP1
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICERED TO LISSERC OF ICE301
           MOVE ICEREV TO LISSERH OF ICE301
           CALL 'ICEMCOD3' USING ICE201 ICE301 ICE301
```

13

```
           MOVE ICECRS TO LISSERL OF ICE301
           PERFORM ICESBR-TEST
           MOVE LISSERC OF ICE101
             TO LISSERC OF ICE301
           MOVE LISSERH OF ICE101
             TO LISSERH OF ICE301
           CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301
           MOVE ICE-CATO-1  TO LISSER OF WORKSV.
       *   MOVE LISCOM OF WORKSV TO LISCOMI OF ICE001 <SWI>.
           MOVE SPACES TO ICESBR
           MOVE '01' TO ICE-CASBRCOD
           MOVE LISCOM OF WORKSV TO ICE-CA01DAT
           MOVE USRLAN TO ICE-CA01LAN
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICE-CATO-4  TO LISCOMI OF ICE001.
       *   MOVE LISCAD OF WORKSV TO LISCADI OF ICE001 <SWI>.
           MOVE SPACES TO ICESBR
           MOVE '01' TO ICE-CASBRCOD
           MOVE LISCAD OF WORKSV TO ICE-CA01DAT
           MOVE USRLAN TO ICE-CA01LAN
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICE-CATO-4  TO LISCADI OF ICE001.
       *   MOVE LISSER OF WORKSV TO LISSERI OF ICE001 <SWI>.
           MOVE SPACES TO ICESBR
           MOVE '01' TO ICE-CASBRCOD
           MOVE LISSER OF WORKSV TO ICE-CA01DAT
           MOVE USRLAN TO ICE-CA01LAN
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICE-CATO-4  TO LISSERI OF ICE001.
           GO TO ICE.

       ICE001-E.
           MOVE 'Y' TO ICEETR.
           GO TO ICE.
```

```
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
      *                    FIN DE TRANSACTION                           *
      *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
       ICETTR.
           IF AJOUT
              PERFORM FILL-FILE
      *       CALL 'ICEFIC' USING ZWRITE BZZD20.
              MOVE ZWRITE      TO ICEFCT
              MOVE BZZD20      TO ICEFILEZON
              MOVE SPACES      TO ICETYP
              ADD  110         TO LONENR-SAVE
              MOVE 'ICEDATAB' TO ICENAM
              MOVE LONENR-SAVE TO ICE-1
              CALL 'ICELNK' USING ICENAM
                                  BZZD20
                                  ICE-1
                                  ICECWAA
                                  DFHEIBLK
                                  ICEDSD
                                  ICETCTUA
              PERFORM ICE-TEST.
           IF SUPPR
      *       CALL 'ICEFIC' USING ZDELETE BZZD20.
              MOVE ZDELETE     TO ICEFCT
              MOVE BZZD20      TO ICEFILEZON
              MOVE SPACES      TO ICETYP
              ADD  110         TO LONENR-SAVE
              MOVE 'ICEDATAB' TO ICENAM
              MOVE LONENR-SAVE TO ICE-1
              CALL 'ICELNK' USING ICENAM
                                  BZZD20
                                  ICE-1
                                  ICECWAA
                                  DFHEIBLK
                                  ICEDSD
                                  ICETCTUA
              PERFORM ICE-TEST.
           IF MODIF
      *       CALL 'ICEFIC' USING ZDELETE BZZD20
              MOVE ZDELETE     TO ICEFCT
              MOVE BZZD20      TO ICEFILEZON
              MOVE SPACES      TO ICETYP
              ADD  110         TO LONENR-SAVE
              MOVE 'ICEDATAB' TO ICENAM
              MOVE LONENR-SAVE TO ICE-1
              CALL 'ICELNK' USING ICENAM
                                  BZZD20
                                  ICE-1
                                  ICECWAA
                                  DFHEIBLK
                                  ICEDSD
                                  ICETCTUA
              PERFORM ICE-TEST
              PERFORM FILL-FILE
```

18

```
*       CALL 'ICEFIC' USING ZWRITE BZZD20
        MOVE ZWRITE      TO ICEFCT
        MOVE BZZD20      TO ICEFILEZON
        MOVE SPACES      TO ICETYP
        ADD  110         TO LONENR-SAVE
        MOVE 'ICEDATAB' TO ICENAM
        MOVE LONENR-SAVE TO ICE-1
        CALL 'ICELNK' USING ICENAM
                            BZZD20
                            ICE-1
                            ICECWAA
                            DFHEIBLK
                            ICEDSD
                            ICETCTUA
        PERFORM ICE-TEST
        IF RETCOD OF BZZD20 NOT = 0
           MOVE 'ZXXX0099' TO ICEMES
*          ENREGISTREMENT DOUBLE (ABEND)
           MOVE 'BZZD20' TO ICEL24
           PERFORM ICE.
     MOVE 'Y'                TO ICEEPR.
     GO TO ICE.

 ICETPR.
     GO TO ICE.

*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*
*                  P E R F O R M S                                      *
*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*

 PRO-SCR1.
*    MOVE ICEASK TO EXTENSI OF ICE001.
     MOVE ICEASK TO EXTENSA OF ICE101
     MOVE ICEASK TO EXTENSA OF ICE301.
*    MOVE ICEYEL TO EXTENSI OF ICE001.
     MOVE ICEYEL TO EXTENSC OF ICE101
     MOVE ICEYEL TO EXTENSC OF ICE301.
*    MOVE ICEASK TO NOMALPI OF ICE001.
     MOVE ICEASK TO NOMALPA OF ICE101
     MOVE ICEASK TO NOMALPA OF ICE301.
*    MOVE ICEYEL TO NOMALPI OF ICE001.
     MOVE ICEYEL TO NOMALPC OF ICE101
     MOVE ICEYEL TO NOMALPC OF ICE301.
*    MOVE ICEASK TO NOMTITI OF ICE001.
     MOVE ICEASK TO NOMTITA OF ICE101
     MOVE ICEASK TO NOMTITA OF ICE301.
*    MOVE ICEYEL TO NOMTITI OF ICE001.
     MOVE ICEYEL TO NOMTITC OF ICE101
     MOVE ICEYEL TO NOMTITC OF ICE301.
*    MOVE ICEASK TO SERVICI OF ICE001.
     MOVE ICEASK TO SERVICA OF ICE101
     MOVE ICEASK TO SERVICA OF ICE301.
*    MOVE ICEYEL TO SERVICI OF ICE001.
     MOVE ICEYEL TO SERVICC OF ICE101
     MOVE ICEYEL TO SERVICC OF ICE301.
```

16

```
      *     MOVE ICEASK TO FONCTII OF ICE001.
            MOVE ICEASK TO FONCTIA OF ICE101
            MOVE ICEASK TO FONCTIA OF ICE301.
      *     MOVE ICEYEL TO FONCTII OF ICE001.
            MOVE ICEYEL TO FONCTIC OF ICE101
            MOVE ICEYEL TO FONCTIC OF ICE301.
      *     MOVE ICEASK TO RECPERI OF ICE001.
            MOVE ICEASK TO RECPERA OF ICE101
            MOVE ICEASK TO RECPERA OF ICE301.
      *     MOVE ICEYEL TO RECPERI OF ICE001.
            MOVE ICEYEL TO RECPERC OF ICE101
            MOVE ICEYEL TO RECPERC OF ICE301.
      *     MOVE ICEASK TO LISCOMI OF ICE001.
            MOVE ICEASK TO LISCOMA OF ICE101
            MOVE ICEASK TO LISCOMA OF ICE301.
      *     MOVE ICEYEL TO LISCOMI OF ICE001.
            MOVE ICEYEL TO LISCOMC OF ICE101
            MOVE ICEYEL TO LISCOMC OF ICE301.
      *     MOVE ICEASK TO LISCADI OF ICE001.
            MOVE ICEASK TO LISCADA OF ICE101
            MOVE ICEASK TO LISCADA OF ICE301.
      *     MOVE ICEYEL TO LISCADI OF ICE001.
            MOVE ICEYEL TO LISCADC OF ICE101
            MOVE ICEYEL TO LISCADC OF ICE301.
      *     MOVE ICEASK TO LISSERI OF ICE001.
            MOVE ICEASK TO LISSERA OF ICE101
            MOVE ICEASK TO LISSERA OF ICE301.
      *     MOVE ICEYEL TO LISSERI OF ICE001.
            MOVE ICEYEL TO LISSERC OF ICE101
            MOVE ICEYEL TO LISSERC OF ICE301.
      *     MOVE ICEASK TO SUPPREI OF ICE001.
            MOVE ICEASK TO SUPPREA OF ICE101
            MOVE ICEASK TO SUPPREA OF ICE301.
      *     MOVE ICEYEL TO SUPPREI OF ICE001.
            MOVE ICEYEL TO SUPPREC OF ICE101
            MOVE ICEYEL TO SUPPREC OF ICE301.

       FILL-SCR1.
            MOVE EXTEN3 OF SAVZON              TO EXTENSI OF ICE001.
            MOVE MATRIC OF SAVZON              TO MATRICI OF ICE001.
            MOVE NOMALP OF CLEF04 OF BZZD20 TO NOMALPI OF ICE001.
            MOVE SERVIC OF CLEF03 OF BZZD20 TO SERVICI OF ICE001.
            MOVE NOMTIT             OF BZZD20 TO NOMTITI OF ICE001.
            MOVE '01' TO NUMTAB OF BZZD01.
            MOVE SERVIC OF CLEF03 OF BZZD20 TO CLETAB OF BZZD01.
      *     CALL 'ICEFIC' USING ZREADIND BZZD01.
            MOVE ZREADIND    TO ICEFCT
            MOVE BZZD01      TO ICEFILEZON
            MOVE ZLIVE       TO ICETYP
            ADD  110         TO LONENR-SAVE
            MOVE 'ICEDATAB' TO ICENAM
            MOVE LONENR-SAVE TO ICE-1
            CALL 'ICELNK' USING ICENAM
                                BZZD01
                                ICE-1
```

17

```
                                    ICECWAA
                                    DFHEIBLK
                                    ICEDSD
                                    ICETCTUA
           PERFORM ICE-TEST.
           IF RETCOD OF BZZD01  = 0
              MOVE ABGSER OF BZZD01 TO ABGSERI OF ICE001
           ELSE
              MOVE SPACES           TO ABGSERI OF ICE001.
           MOVE FONCTI            OF BZZD20 TO FONCTII OF ICE001.
           MOVE RECPER            OF BZZD20 TO RECPERI OF ICE001.
       *   MOVE LISCOM OF BZZD20 TO LISCOMI OF ICE001 <SWI>.
           MOVE SPACES TO ICESBR
           MOVE '01' TO ICE-CASBRCOD
           MOVE LISCOM OF BZZD20 TO ICE-CA01DAT
           MOVE USRLAN TO ICE-CA01LAN
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICE-CATO-4  TO LISCOMI OF ICE001.
       *   MOVE LISCAD OF BZZD20 TO LISCADI OF ICE001 <SWI>.
           MOVE SPACES TO ICESBR
           MOVE '01' TO ICE-CASBRCOD
           MOVE LISCAD OF BZZD20 TO ICE-CA01DAT
           MOVE USRLAN TO ICE-CA01LAN
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICE-CATO-4  TO LISCADI OF ICE001.
       *   MOVE LISSER OF BZZD20 TO LISSERI OF ICE001 <SWI>.
           MOVE SPACES TO ICESBR
           MOVE '01' TO ICE-CASBRCOD
           MOVE LISSER OF BZZD20 TO ICE-CA01DAT
           MOVE USRLAN TO ICE-CA01LAN
           MOVE 'ICESBRB' TO ICENAM
           MOVE +174 TO ICE-1
           CALL 'ICELNK' USING ICENAM
                               ICESBR
                               ICE-1
                               ICECWAA
                               DFHEIBLK
                               ICEDSR
           MOVE ICE-CATO-4  TO LISSERI OF ICE001.
           MOVE 'N'                TO SUPPREI OF ICE001.
```

```
       FILL-FILE.
           MOVE EXTENS  OF WORKSV TO EXTENS OF CLEFIC OF BZZD20
                                     EXTENS OF CLEF02 OF BZZD20
                                     EXTENS OF CLEF03 OF BZZD20.
           MOVE NOMALPI OF ICE001 TO NOMALP OF CLEF02 OF BZZD20
                                     NOMALP OF CLEF04 OF BZZD20.
           MOVE SERVICI OF ICE001 TO SERVIC OF CLEF03 OF BZZD20.
           MOVE MATRICI OF ICE001 TO MATRIC OF CLEFIC OF BZZD20.
           MOVE FONCTII OF ICE001 TO FONCTI OF BZZD20.
           MOVE RECPERI OF ICE001 TO RECPER OF BZZD20.
           MOVE LISCOM  OF WORKSV TO LISCOM OF BZZD20.
           MOVE LISCAD  OF WORKSV TO LISCAD OF BZZD20.
           MOVE LISSER  OF WORKSV TO LISSER OF BZZD20.
           MOVE NOMTITI OF ICE001 TO NOMTIT OF BZZD20.
       ICETSW-E.
           GO TO ICE.
       ICEBRA.
           IF ICECUR = 1
              GO TO ICE001-I ICE001-T ICE001-E
                    DEPENDING ON ICEACT.
       ICE-MOVE-TO-A.
           GO TO ICEMTO-001
                 DEPENDING ON ICECUR.
       ICEMTO-001.
           IF ICEBATCH = 'Y' AND ICECST = 4
              MOVE BATBUF OF ICEITF TO ICE001.
           CALL 'ICEMC003' USING ICE201 ICE301 ICE001 ICE301.
           MOVE ICE301 TO ICEBUF.
           GO TO ICE-MOVE-TO-Z.
       ICE-MOVE-TO-Z.
       ICE-MOVE-SCL-A.
           GO TO ICEMSC-001
                 DEPENDING ON ICECUR.
       ICEMSC-001.
           MOVE ICEBUF TO ICE301.
           PERFORM ICEMCO-301-001.
           CALL 'ICEMC031' USING ICE201 ICE301 ICE301 ICE101.
           GO TO ICE-MOVE-SCL-Z.
       ICE-MOVE-SCL-Z.
       ICE-MOVE-FROM-A.
           GO TO ICEMFR-001
                 DEPENDING ON ICECUR.
       ICEMFR-001.
           MOVE ICEBUF TO ICE301.
           PERFORM ICEMCO-101-301.
           CALL 'ICEMCOPC' USING ICE201 ICE301 ICE301.
           MOVE ICE301 TO ICEBUF.
           MOVE ICE001 TO ICEBUS.
           PERFORM ICEMCO-301-001.
           IF ICE001 NOT = ICEBUS
              MOVE 'Y' TO ICEBUD.
           GO TO ICE-MOVE-FROM-Z.
       ICE-MOVE-FROM-Z.
       ICEMCO-101-301.
           CALL 'ICEMC013' USING ICE201 ICE301 ICE101 ICE301.
```

```
           ICEMCO-301-001.
               CALL 'ICEMCO30' USING ICE201 ICE301 ICE301 ICE001.
           *     COPY ICEMTP SUPPRESS.
       CBL XOPTS(CICS DEBUG LANGLVL(2))                                  I
            ICE.                                                         I
           *    ICEMTP : CODING PROVIDED AFTER USER PROGRAM              I
               IF (ICENUM = ALL SPACES)                                  I
                  MOVE 1 TO ICEBCH                                       I
                  GO TO LINK-ICEMTP.                                     I
               MOVE 16 TO ICEBCH.                                        I
               IF ICECST = 5                                             I
                  PERFORM ICE-MOVE-TO-A THRU ICE-MOVE-TO-Z.              I
               PERFORM LINK-ICEMTP.                                      I
               IF ICEBCH NOT = 99                                        I
                  GO TO ICE-RETURN-ICEMTP.                               I
           *  GO BACK HERE FOR THE 'PERFORM ICE' REACHING THIS POINT      I
           *  START PROGRAM                                              I
            ICE-START.                                                   I
               EXEC CICS ADDRESS TCTUA (ICETRM)                          I
                                 TWA   (ICE-TWA) END-EXEC.               I
               MOVE EIBTRNID  TO ICEAPP.                                 I
               EXEC CICS HANDLE ABEND LABEL(ICE-ABEND) END-EXEC.         I
               CALL 'ICEADR' USING ICEREC ICEREC-END ICEAWS ICELT2.      I
               MOVE 9 TO ICEBCH.                                         I
           LINK-ICEMTP.                                                  I
               MOVE 'ICEMTXB' TO ICENAM.                                 I
               MOVE 4089 TO ICE-1.                                       I
               CALL 'ICELNK' USING ICENAM ICESUP ICE-1 ICECWAA DFHEIBLK  I
                                   ICEDSX ICETWA ICETCTUA.               I
               IF ICECWAA = 99999999                                     I
                  EXEC CICS ABEND ABCODE('ICCT') CANCEL END-EXEC.        I
           ICE-RETURN-ICEMTP.                                            I
               GO TO     ICEIPR      ICEITR     ICEBRA                   I
                         ICETTR      ICETPR     ICE-EXIT                 I
                         ICE-SEND    ICE-EXIT   ICETSW-E                 I
                         ICE-RETURN  ICE-AFTER  ICE-MOVE-FR              I
                         ICE-EXIT                                        I
               DEPENDING ON ICEBCH.                                      I
           *  BEFORE SENDING A MAP : FILL BUFFER AND MAP NAME            I
            ICE-SEND.                                                    I
               MOVE MAPICE (ICECUR)  TO ICEPMA.                          I
               PERFORM ICE-MOVE-TO-A THRU ICE-MOVE-TO-Z.                 I
               MOVE 11      TO ICEBCH.                                   I
               GO TO  LINK-ICEMTP.                                       I
           *  FILL WS MAP AND ICEBUD POSITIONNING                        I
            ICE-MOVE-FR.                                                 I
               PERFORM ICE-MOVE-FROM-A  THRU ICE-MOVE-FROM-Z.            I
               IF ICENUM NOT = SPACES                                    I
                  GO TO ICE-SEND.                                        I
               MOVE 13  TO ICEBCH.                                       I
               GO TO  LINK-ICEMTP.                                       I
           *  AFTER SENDING A PAGED MAP                                  I
            ICE-AFTER.                                                   I
               PERFORM ICE-MOVE-SCL-A THRU ICE-MOVE-SCL-Z.               I
               MOVE 5  TO ICEBCH.                                        I
```

20.

```
           GO TO LINK-ICEMTP.
       *  ERROR CICS
        ICE-ERROR.
           MOVE 6 TO ICEBCH.
           GO TO LINK-ICEMTP.
       *  CICS ABEND
        ICE-ABEND.
           EXEC CICS HANDLE ABEND CANCEL END-EXEC.
           MOVE 7 TO ICEBCH.
           GO TO LINK-ICEMTP.
       *
        ICE-TASK-SWITCH.
           IF ICETSD NOT = 'Y'
              MOVE 251  TO ICE-1
              EXEC CICS LINK PROGRAM ('IC05')
                             COMMAREA (ICESUI)
                             LENGTH(ICE-1)
                             END-EXEC
              MOVE 'Y'          TO ICETSX.
        ICETIME.
           EXEC CICS ASKTIME END-EXEC.
           MOVE EIBTIME TO ICETIM9.
        ICEDELAY.
           IF ICEDEL > 0 AND ICEDEL < 60
              EXEC CICS DELAY INTERVAL(ICEDEL) END-EXEC.
       *  UNLOCK ALL FILES
        ICE-CLEAR.
           MOVE 'ICECLRB' TO ICENAM.
           CALL 'ICELNK' USING ICENAM ICE-1  ICE-1 ICECWAA DFHEIBLK
                               ICEDSD ICETCTUA.
       *  TEST AFTER LINK ICEDATA
        ICE-TEST.
           IF ICEFLG = 'Y'
              GO TO ICE-ERROR.
       *  TEST AFTER LINK ICESBR
        ICESBR-TEST.
           IF ICERET OF ICESBR NOT = '00'
              MOVE 'ICE '               TO ICEGEN
              MOVE ICESBR               TO ICENUM
              PERFORM ICE.
       *     TEST AFTER SUBROUTINE CALL
        ICE-CALL-TEST.
           IF ICEFLGT = 'A'
              GO TO ICE-ABEND.
           IF ICEFLG = 'Y'
              GO TO ICE-ERROR.
        ICE-EXIT.
           EXEC CICS RETURN END-EXEC.
        ICE-RETURN.
           EXEC CICS RETURN TRANSID (EIBTRNID) END-EXEC.
```

NOTES

01) MAINFRAME : Ordinateur central : à la Belgolaise un IBM 4381 [Cfr. II.2].

02) PC : Personal Computer = ordinateur personnel

03) ICE : L'environnement de développement logiciel transactionnnel mis au point et utilisé systématiquement par les programmeurs de la Banque [Cfr III]

04) FBSS : Financial Branch System Services" [Cfr IV - 2.1.b.]

05) DOS : Disk Operating System" sur PC (Microsoft) [Cfr IV - 2.1.b.]

06) UNIX : Système d'exploitation pour mini-ordinateurs

07) BANQUE BELGO-ZAIROISE ou la BELGOLAISE, ou la BBZ : [Cfr I - 1]

08) ICE-PC : La version PC de l'environnement de développement logiciel ICE [3] [Cfr IV]

09) LAN : "Local Area Network" = réseau local de PC

10) CREDITS DOCUMENTAIRES : [Cfr I - 2]

11) SERVICE VISA : [Cfr Annexe 1]

12) MIPS : Million d'instructions par seconde

13) OS : Système d'exploitation

14) ABB : Association Belge des Banques

15) TOKEN RING : LAN IBM (réseau local à jeton)

16) SGBD : Système de Gestion des Bases de Données

17) STI : Système de traitement de l'information Batch c'est le moniteur de contrôle de modules J1, J2 et J3 [Cfr II]

18) STICS : Système de traitement de l'information (ONLINE sous CICS)

19) PIPE UNIX : Fonctionnalité de UNIX permettant de diriger l'output d'une commande vers l'input de la suivante

20) DBASE : Logiciel de base de données sur PC (Ashton Tate)

21) LAN LOCAL AREA NETWORK : Réseau local de PC

22) BOITE A OUTILS : [Cfr III - 5.1.]

23) B-LINE : [Cfr I - 5.3.]

24) MVS : Multiple Virtual Storage" (Système d'exploitation IBM)

25) CICS : Utilitaire de MVS facilitant la gestion des écrans et les accès fichiers

26) DL1 : Un SGBD particulier

27) DEAD LOCK : Problème de base de données. Lorsque deux enregistrements sont verrouillés et que chacun attend que l'autre soit déverrouillé pour se déverrouiller

28) JCL : Job Control Langage (Langage de contrôle des jobs)

29) CPU : Central Process Unit (unité centrale de calcul)

30) ICEFIC : (Interface d'accès aux données) [Cfr III - 7.1.a.]

31) ROLL BACK : Par lequel un fichier retrouve son état initial [Cfr III- 6.8.]

32) CPU : Central Unit Process

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[AB89] Rapport de l'Association Belge des Banques (1988).

[BG56] Production of large computer programs, H. D. Benington dans "Proc. ONR Symp. Advanced Programming Methods for Digital Computers", pp. 15-27, June (1956).

[BL79] Thinkin Ahead : Communications Technology - for Better or for Worse, D. Bell, dans "Harvard Business Review", May-June, pp. 20-42 (1979).

[BL88] Documents internes de la Belgolaise (1988).

[BM84] Software engineering economics, B. W. Boehm dans "IEEE transaction on software engineering", vol. SE-10 N°. 1, January (1984).

[BO89] Conception assistée des applications informatiques, 1. étude d'opportunité et analyse conceptuelle, F. Bodart et Pigneur, Masson (1989).

[CK85] IS Redraws Competitive Boundaries, J. I. Cash and B. R. Konsynski, dans "Harvard Business Review", March-April, pp. 134-42 (1985).

[DATE] An Introduction to Data Base System (Vol. I et II), par C.J. Date (IBM Corporation), Editeur : Addison-Wesley Publishing Compagny, ISBN 0-201-14474-3.

[DK68] A constructive approach to the problem of program correctnes, E. W. Dijkstra, BIT, 8, 174-86, (1968).

[DM76] An Introduction to the Programmer's Workbench, T.A. Dolotta et J.R. Mashey dans "Proc. 2d Intl. Conf. Software Engineering", Oct. (1976).

[DS68] Go To Statement Considered Harmful, E. W. Dijkstra, dans "Communication of the ACM" vol. 11 N° 3 Mach (1968).

[ER88] Information Management : Some Strategic Reflections, Michael J. Earl, Oxford Institute of Information Management, Templeton College, Oxford; dans "IT and Stategy, Reflections and Direction" de Michael J. Earl, Oxford Institute of Information Management, Templeton College, Oxford.

[FBSS] FBSS Version 2.1 Function and Facilities, May/89, Editeur : IBM.

[GT70] Designing Systems Programs, R. Gauthier and S. Pont, Prentice-Hall, Englewood Cliffs, N.J., (1970).

BIBLIOGRAPHIE

[IV77] The Programmer's Workbench - A Machine for Software Development, E.L. Ivie in Comm. ACM. 20 (10), pp. 746-753, Oct. (1977).

[JV88] Creating Competitive Advantage with Inter organisational Information System, H. R. Johnston et M.R. Vitale, MIS Quaterly (1988) p. 161.

[KN86] Competitive in time : Using Telecommunications for Competitive Advantage, P. G. W. Keen, Ballinger, Cambridge, Mass (1986).

[KR81] Software Tools in Pascal, B.M. Kernigham et P.J. Plauger, Addison-Wesley Reading, Mass. (1981).

[LB89] La fonction informatique : une fonction en évolution ?, C. Lobet, dans "Journal de réflection sur l'informatique" N° 12 - janvier 1989, F.N.D.P. Namur.

[LS89] La fonction informatique dans l'entreprise, R. Lesuisse, dans "Journal de réflection sur l'informatique" N° 12 - janvier 1989, F.N.D.P. Namur.

[LV65] Applying Organisational Change in Industry : Structural,Tecnological and Humanistic approaches, H. S. Leavitt dans I.J.G. March (édit), Handbook of Organisation, Chicago, Rand, Mc Nally (1965).

[NL79] Managing the crisis in data processing, R. L. Nolan, Harvard Business Review, pp. 115 à 128, (1979).

[PRMS] IBM Financial Branch System Service Version 2.0 Release 1.0 Programmer's Reference Manual, May/89, Editeur : IBM.

[PR72] On the Criteria To Be Used in Decomposing Systems into Modules, D.L. Parnas, Carnegie-Mellon University, dans "Communication of ACM" Vol. 15 N° 12 December 1972.

[PR77] Use of Abstract Interfaces in the Development of Software for Embedded Computer System, D.L. Parnas, NRL Report 8047, Manuscript submitted in April (1977).

[PR79] Stratégie : analyser votre industrie, 1979, Harvard : L'expansion pp. 100-111.

[RF80] Software Development Tools, W.E. Riddle et R.E. Fairley, Springer Verlag, Berlin (1980).

[SE76] COBOL, M.A. Jackson, Michael Jackson System Limited, Pinner, Middlesex, dans "Software Engineering : Proceeding of a Symposium held at The Queen's University of Belfast", Edited by R.H. Perrott Department of Computer Science The Queen's University of Belfast, Belfast, N. Ireland (1976).

BIBLIOGRAPHIE

[SG81] Special Issue on Programming Environments, Sigsoft dans "ACM Software Engineering Notes", 6 (4), Aug. (1981).

[SM82] Software Engineering (second edition), I. Sommerville, Editeur Addison-Wesley Publishing Company (1982).

[VLWD] Les outils d'aide au développement de logiciels, A. van Lamweerde.

[WG79] Research Directions in Software Technology, P. Wegner, MIT Press, Cambridge, Mass (1979).

[WL76] Expérimentation du générateur ATLAS, R. Weil dans "Service Technique Informatique", mai (1976).

[WL89] Etude Professeur Wilkin U.L.B. (1989).

[WR71] Program development by stepwise refinement, N. Wirth dans "Communication of ACM", 14 (4), 221-7 (1971).