ANNEXE **T**

PRECISIONS

TECHNIQUES

.M1.ANNEXE T-1

SOURCE	*L'informatique et nous*, op. cit., p.

COMMENTAIRE	Présentation de l'informatique en « presse-bouton ». Extrait du chapitre premier de notre ouvrage sur l'informatique à taille humaine (paru en janvier 1985) [1].

.M2.L'informatique en « presse-bouton »

.M3.La micro-informatique pour quoi faire ?

Les ordinateurs font maintenant partie de la vie courante, cependant, leurs utilisations réelles demeurent encore en partie peu connues. C'est pourquoi nous allons en premier lieux rappeler leurs principaux usages, aussi bien dans le cadre d'une association, que d'une municipalité, ou plus généralement d'une petite entreprise.
Mais avant cela, il nous semble utile de nous interroger sur la notion de "système fermé et autonome", intégralement automatique, n'exigeant aucune connaissance de caractère informatique... C'est ce que l'on appelle communément le système "presse-bouton".

.M3.LE PRESSE-BOUTON - L'UTILISATION DE LOGICIELS COMMERCIAUX.

De toutes les utilisations des ordinateurs, celle qui consiste à avoir recours à des systèmes totalement automatiques (des "boites noires"), installés par des spécialistes et remis "clés en main" aux utilisateurs apparait souvent comme la plus sure, au moins pour des problèmes simples, bien définis, et fortement répétitifs. Mais elle est aussi la plus onéreuse, ou pour être plus précis, c'est celle qui exige les dépenses immédiates les plus importantes, même si à l'usage son coût redevient moins exorbitant.
Mais d'abord, qu'appelle-t-on un usage "presse-bouton" des ordinateurs?
On peut dire qu'il s'agit d'une informatique "par le haut", qui ne fait aucunement appel à l'intelligence, à la "ressource humaine" de ses utilisateurs. Il s'agit d'une sorte de "taylorisme" qui vise d'abord à augmenter la productivité. On l'a vu avec les machines-outils, puis les premiers robots industriels. Par la suite, le "presse-bouton" s'est attaché également à rendre la vie plus facile, les taches moins pénibles, moins salissantes, moins absorbantes ou carrément moins dangereuses lorsqu'il y

---

1 On notera, au titre de l'anecdote technique, que ce texte a été tapé sur un ordinateur Apple 2 en juillet/août 1984, puis photocomposé pour notre ouvrage chez ESF/EME et ensuite « récupéré » sur Macintosh pour la présente thèse. Ceci montre à l'évidence, que le plus important dans toute approche de l'informatique, ce sont bien les *données* qu'il faut pouvoir réexploiter plusieurs années plus tard.

a risque d'accident.
Malgré de nombreux aspects "positifs", l'emploi du "presse-bouton" ne semble pas recueillir beaucoup de succès parmi les personnes directement intéressées. On peut y voir deux raisons principales : la première tient à la crainte pas toujours dénuée de fondement selon laquelle la machine asservira l'être humain à sa cadence, à son rythme (cf. Les Temps Modernes)... La seconde tient au sentiment souvent exact de se sentir "dépossédé" de son travail, de sa qualification, et plus généralement, de son "utilité sociale". Ce dernier argument ne concernant évidemment que les métiers déjà relativement qualifiés (ouvriers professionnels) ou croyant l'être davantage qu'il ne l'est réellement (une partie des personnels de bureau).
Pour en finir avec cette question, à nos yeux fondamentale, notamment à la lumière des nouveautés technologiques qui s'annoncent, nous distinguerons très rapidement les principaux avantages du "presse-bouton", de ses plus notables défauts :

LES AVANTAGES

- Une productivité assez fortement augmentée, au moins dans le cas d'une "informatisation réussie";

- Une diminution en volume du personnel d'exécution nécessaire (on retrouve-là le couplet maintenant connu sur l'extinction progressive des industries de pure main d'oeuvre, telles que les civilisations industrielles les ont développées);

- Une diminution de la pénibilité des principales taches (ceci est surtout vrai pour l'instant dans le secteur secondaire avec l'arrivée de la robotique);

- Une simplification de la formation professionnelle (d'ou une réduction de son coût...).

LES INCONVENIENTS

- Un risque de "dérapage" des cadences de production, pliant encore davantage l'être humain au rythme des machines (on en trouve un exemple avec l'accroissement du travail de nuit, et la tendance actuelle à étendre le travail posté aux emplois de bureau nouvellement informatisées (#)...);

- Un risque de "dépersonnalisation" du travail amenant à terme à un désintérêt total des personnels pour ce qu'ils font (on en observe déjà quelques prémices dans certains bureaux ou ateliers...);

- Un risque de "déresponsabilisation" se traduisant par le déjà célèbre : "c'est la faute à l'ordinateur..."

- Un manque d'adaptabilité des machines à des taches nouvelles (ce qui fait qu'on peut les qualifier de machines "bêtes", par opposition aux "systèmes intelligents");

- Des économies parfois illusoires quant au coût réel de production (investissements très lourds et figés dans le temps, c'est à dire tributaires du moindre ralentissement des activités couvertes);

- Une réduction à terme du dynamisme social figeant les citoyens dans des postes immuables, pour des activités répétitives.

.M3.LE PRESSE-BOUTON, OUI... MAIS A CONDITION DE PARTICIPER AU REGLAGE DU BOUTON ! [1]...

On l'aura compris, l'ordinateur en "presse-bouton" ne nous semble pas être la panacée informatique, et les associations, comme les municipalités répondraient mal à l'une de leurs missions essentielles de découverte et de familiarisation vis à vis des nouveaux moyens de communication, en s'engageant dans cette voie un peu étroite. Tout au plus devrait-on en retenir les gains de productivité et les baisses de pénibilité des taches.
La bonne solution nous semblerait être celle qui privilégierait la prise en compte des ressources des individus (à condition de les former convenablement), et leur accorderait une part suffisante de responsabilité et d'autonomie dans l'organisation et l'exécution de leurs taches. Naturellement, les associations, comme les municipalités et certaines PME/PMI restant redevables de leurs actes, de leur gestion et de leurs services devant leurs mandants, continueraient de rechercher la meilleure efficacité (ou "productivité" possible, compte tenu des objectifs précédemment définis, dans une perspective de développement maîtrisé des nouveaux moyens de communication.

1 On peut rapprocher cette réflexion sur les « boutons » avec les commandes du logiciel « Hypercard » qui recourt justement sans cesse aux boutons pour paramétrer des applications personnelles.

.M1.ANNEXE T-2

SOURCE	Travaux personnels

COMMENTAIRE	Présentation rapide des logiciels que nous avons employés pour la réalisation de la présente thèse.

.M2.Les logiciels utilisés

Traitement de texte :

MacWrite (Apple) : Jusqu'en septembre 1987.
Word 3 (Microsoft) : Depuis septembre 1987 - Utilisation continuelle.

Traitement graphique :

MacPaint (Apple) : tous les dessins contenant des courbes non régulières. Récupération et traitement des copies d'écran (FileVision et MacSpin par exemple).
MacDraw (Apple) : tous les autres dessins et schémas. Récupération et traitement de graphiques issus d'autres logiciels.
Superpaint : Récupération et traitement de schémas « pleine-page » [1].
Mac Bilboard : Récupération et traitement de schémas « pleine-page ».

Accessoire « **Art Grabber** » permettant de récupérer des dessins ou des schémas directement dans un traitement de texte.

Grapheur du tableur « **Excel** » (Microsoft) : pour tous les graphiques.

Modélisations mathématiques et calculs sur tableur :

Excel (Microsoft) : Toutes les modélisations. Utilisation continue. Usages supplémentaires pour récupérer diverses données « anciennes »

Gestion des fichiers bibliographiques et des fichiers de « mots-clés » :

ABC-Base (ACI): Paramétrage de la gestion de la bibliographie (recherches multicritères et par mots-clés). Développement des maquettes citées dans les annexes.
Quatrième Dimension (**4D**) : Mêmes usages, avec des fonctions sup-

1 Nous alternions l'usage de ces deux logiciels en fonction de la quantité de mémoire disponible (1 méga-octet de mémoire vive). Configuration de travail : Macintosh Plus (1 MO, lecteur de disquettes interne de 800 KO). Jusqu'en septembre 1987, second lecteur de disquettes externe de 400 KO. A partir de septembre 1987, disque dur de 32 MO (Crex). Modem du minitel.

plémentaires.

Gestion de fichiers graphiques :

FileVision : Modélisation des appartenances associatives (étude des liens). Modélisation du partenariat associatif (étude des liens).
Hypercard : Mêmes fonctions et préparation de la soutenance.

Modélisation en trois dimensions :

Mac Spin (Central Point) **:** Typologie associative en 3D. Regroupements poly-catégoriels. Variation des coefficients de regroupement.

Excel (Microsoft) **:** Calculs de tendances, graphiques divers. Traitement de questionnaires. Profils-types.

Mise au point des deux mini systèmes-experts [1] *:*

Turbo SE : (Ph. Larvet) Mise au point de deux mini systèmes-experts (environ 200 règles chacun).
Reconnaissance automatique des « profils médiatiques ».
Reconnaissance automatique des genres d'associations.

Connections télématiques :

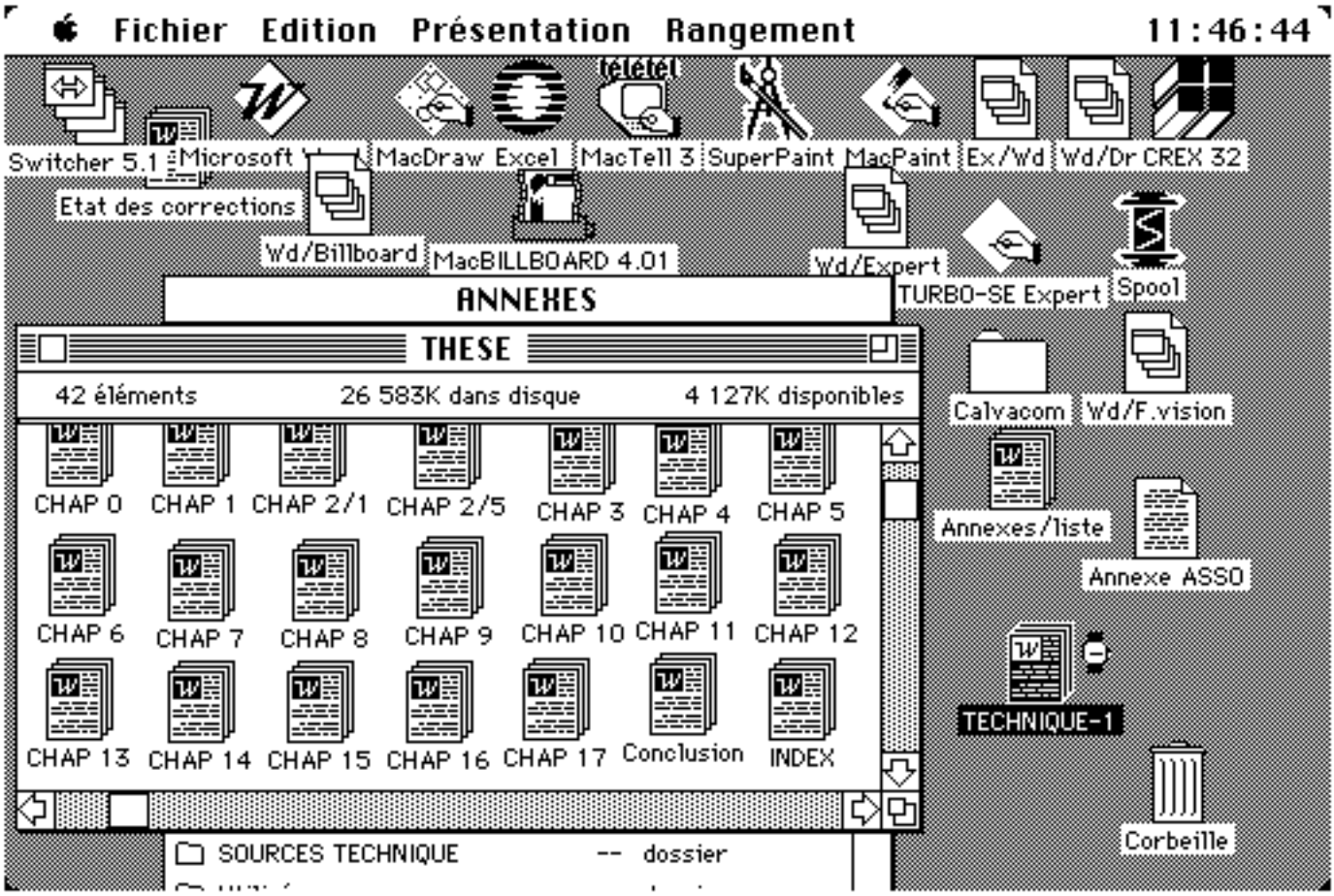
Mac Tell (Hello informatique) **:** Récupération et traitement d'images videotex et d'écrans ASCII.

Mise en page :

Word 3 : Pour la mise en âge « classique »
Page Maker (Aldus) : Pour les effets spéciaux (liste des hypothèses par exemple).

1 Avant de disposer de cet excellent logiciel, nous utilisions « Mac Expert », qui fut le premier système-expert disponible sur Macintosh. Pour des raisons financières évidentes, nous n'avons pu utiliser des systèmes d'ordre 1 ou 1+ comme ceux développés par ACT-informatique.

Illustration du « bureau électronique » du Macintosh lors du début d'une session de travail :



.M1.ANNEXE T-3

SOURCE	Serveur Calvacom - Revues d'informatique.

COMMENTAIRE	Quelques précisions sur l'hypertexte (et les hypermédias).

.M2.L'hypertexte

La notion d'hypertexte se rapporte à *la lecture et à l'écriture non-séquentielle*. Tout à la fois outil pour les auteurs et médias pour les lecteurs, l'hypertexte permet aux premiers de créer des passerelles entre leurs données, d'annoter des textes existants et offre aux seconds de multiples chemins pour accéder à l'information recherchée. Par extension, le terme « **hypermédia** » désigne tout tout système fonctionnant comme l'hypertexte, mais avec des graphiques, des sons, et des images animées.
Un logiciel d'hypertexte permet de « cliquer » sur un mot et d'obtenir les désinences ou acceptions, ou définitions de celui-ci, ainsi que des exemples d'emploi. Tout se passe comme si on disposait « en ligne », c'est-à-dire en direct d'une encyclopédie ou d'une série de dictionnaires. De plus, les liens entre les mots (ou les schémas) ou les dessins peuvent être mémorisés et reproduits.
Les applications éducatives paraissent immenses. Nous les avons soulignées au chapitre

« Avec l'hypermédia, les étudiants sont beaucoup plus actifs qu'en classe traditionnelle. Mieux encore, ils établissent parfois des liaisons auxquelles je n'avais pas pensé. *Ils trouvent de nouvelles manières de connecter entre elles différentes parties de la base de données*. En manipulant eux-mêmes l'information, ils la retiennent beaucoup plus facilement. Car on apprend mieux à travers la découverte. » [1].

Afin de fixer les idées de manière plus concrète, nous allons citer quelques réactions suscitées par le système Hypercard (présenté par Apple en août 1987), qui est la première présentation grand public d'un logiciel de ce genre.

.M3.Extrait d'un article de **Libération** *(capté sur le serveur Calvacom [2]) - 13/08/87*

1 Interview in Revue *Icônes*, n°9, 10/11/1987, p. 46.

2 Et « traité » par nous ensuite (au point de vue orthographique et syntaxique). C'est nous qui soulignons.

693 - « Apple lance la boîte a outils informatique » (48 l.) Alain VACHE [1] (AV10) D 44 - 14 août 7 15h56 (172 lec.) Origine: 689

Plus que la description sommaire du produit, c'est la présentation qui en est faite qui peut retenir l'attention. Il s'appelle Hypercard. C'est le dernier né de chez APPLE. Destiné à la gamme Macintosh, *c'est le premier logiciel qui permette à l'utilisateur d'organiser et de manipuler des informations comme il le souhaite*. Cette révolution de l'informatique arrivera en France en octobre. Apple a-t-il brisé la muraille de Chine que constituent, pour les non informaticiens, les voies impénétrables de la programmation ? Il faudra attendre octobre prochain et Apple-Expo, pour juger, sur pièces, de la taille du nouveau pavé que la firme de Cupertino jette dans la mare de la micro-informatique. Il s'appelle *Hypercard*, c'est un logiciel destiné à la gamme Macintosh. Il est vendu 49 dollars (300F), et il devrait mettre la programmation à la portée de tous. D'ores et déjà, les réactions suscitées aux USA par la présentation d'HyperCard sont des plus flatteuses. Les analystes qui l'ont essayé estiment qu'il s'agit du programme le plus significatif mis sur le marché par Apple depuis le lancement de sa gamme d'ordinateurs Macintosh. Hypercard n'est pas un programme d'application comme par exemple un traitement de texte. C'est un programme qui permet d'organiser et de manipuler des informations comme l'utilisateur le souhaite. Or jusqu'ici, le choix était réduit. Soit vous étiez un as de la programmation et vous écriviez vous même vos logiciels. Dans le cas contraire, vous étiez condamnés au prêt-à-porter, autrement dit aux programmes vendus sur le marché, sans pouvoir modifier leur fonctionnement ; c'est-à-dire la manière dont ils organisent les informations et les données que vous leur confiez.
Avec Hypercard l'utilisateur dispose de cartes, sortes de fiches Bristol informatiques. Sur chaque carte affichée à l'écran, il peut rentrer des notes, dessiner, et ensuite les relier entre elles en faisant des sortes de piles grâce a des boutons. Ces boutons déclenchent des commandes contenues dans un script. Les cartes, les piles de cartes reliées et les boutons constituent une sorte de boîte à la disposition de l'utilisateur. Hypercard permet d'écrire son propre scénario informatique, c'est-à-dire ses propres applications au lieu de dérouler sempiternellement celles écrites par les autres (ce qui d'ailleurs peut être suffisant). Avec Hypercard qui a demandé trois ans de développement à Bill Atkinson [2], son créateur, Apple poursuit la philosophie qui est la sienne. Faire des machines dont l'utilisation est la plus simple possible, orientées vers l'utilisateur. Une philosophie qui, jusqu'à présent a plutôt réussi à la firme de Cupertino qui a imposé, face au monde des micros IBM et compatibles, l'autre standard du marché. Hypercard semble constituer un nouveau palier dans la poursuite de cette stratégie. Le programme est vendu à un très bas prix, avant d'être distribué avec les Macintosh. Apple pense distribuer 1,3 million d'hypercard d'ici la fin de l'année. La firme n'a pas de soucis a se faire: les non-informaticiens sont loin d'être une espèce en voie de disparition."

.M3.Extrait d'un article de* 01 Informatique *(septembre 1987) [3]

716 - HyperCard...Quelques précisions (27 l)- Fréderic RINALDI

---

1 Nom du correspondant qui a envoyé l'article en question.

2 Et auteur de « *Macpaint* » qui révolutionna le « contact » avec un ordinateur.

3 Egalement « récupéré » sur Calvacom et traité par nous ensuite.

(FR10) D 57 - 18 août 87 10h06 (150 lec.)

Développé par Bill Atkinson, HC (Hypercard) est d'abord un outil qui permet à un utilisateur de Macintosh d'organiser et d'utiliser sa machine de façon entièrement personnelle et plus efficace. Là où un système informatique cherche à créer une structure logique entre des éléments qui comportent des champs définis au préalable, *le principe sur lequel est basé HC s'appuie sur le modèle du raisonnement humain et de l'association d'idées*. HC est donc un système non structuré qui manie des objets, du texte, du graphique, des images vidéo, de la musique etc.. On peut grouper des objets de même nature ou natures différentes dans des *piles* ("Stacks"), et à l'écran on dispose d'*outils* (les *boutons*) que l'on peut disposer n'importe où. Au moyen de ces boutons, on peut effectuer toutes sortes de manipulations sur un objet, à l'intérieur d'une pile ou créer des liens entre les objets d'une même pile ou de piles différentes, ou encore créer des liens entre les piles, selon ses propres préférences. Un utilisateur peut donc organiser sa propre structure qui regroupe des objets et des actions selon une logique qui lui est totalement personnelle. HC comporte un langage appelé HyperTalk qui est fortement inspiré du *SmallTalk*. C'est un langage du type non procédural, peu structuré, adapté à la manipulation d'objets.

.M3.Explications d'un "spécialiste de la programmation" [1]

260 - Hypercard... (80 l.) Daniel Ranson (DR10) D 75 - 20 sep 87 21h50

Hypercard n'est pas un langage de programmation, ce n'est pas un compilateur et il ne sert pas à produire des applications StandAlone, et ce n'est pas là un défaut du soft : ce n'est tout simplement pas sa vocation. HyperCard est un langage auteur extrêmement évolué, son concept étant poussé aux dernières limites.
Hypercard est en fait une nouvelle sorte d'application, un environnement de gestion de l'information. Il sert à stocker, représenter, ventiler, lier toute sortes d'informations : texte pur, images, images « scannérisés » [2] ou digitalisées, documents sonores (sons, musique, parole), en bref tout ce que le Mac peut véhiculer comme informations. HyperCard devient alors le support universel de ces infos, dotés de sa propre intelligence quand à leur présentation, leur enchaînement, etc. En plus de cela, HyperCard sait gérer des fichiers d'un maniement très simple, fichiers qui peuvent faire l'objet de recherche sur clés, de tris, de croisement, de fusion, d'extraction, etc... et qui utiliseront le même support que les dessins, textes, etc.. généralement importés d'autres applications.
HyperCard vous permettra de connecter instantanément toutes sortes d'informations, avec une facilité pour l'utilisateur telle que vous n'en avez jamais rêvé. Clic sur la souris, l'info est là (très très vite, croyez-moi). Le langage d'HyperCard (HyperTalk) permet de créer des "scripts", qui sont les instructions utiles au soft pour déterminer comment, dans quel ordre, à la suite de quel événement telle ou telle information doit être représentée.

---

1 Daniel RANSON est un contributeur acharné de Calvacom.

2 C'est-à-dire « numérisées » lors d'un passage par un « scanner ».

.M3.L'hypertexte, un nouveau concept ?

270 - Découvertes... (39 l.) Denis BERARD DB40) D 75 - 27 sep 87 02h45
Evidemment on n'écrira pas comme ça un programme de traitement de texte, mais c'est génial pour tout ce qui touche à la gestion des informations, textes ou images...
Mais ce qui me semble le plus important c'est qu'Hypercard permet de passer à une dimension supérieure : le concept souris/menu déroulants était révolutionnaire à l'époque de la sortie du Mac parce que c'était *une ergonomie basée sur une projection du corps* (mouvement) sur les deux dimensions de l'écran. Hypercard nous emmène dans une dimension supérieure *car c'est une projection en trois (ou plus?) dimensions* : on peut passer d'une pile à l'autre, mais aussi se déplacer en profondeur d'un niveau d'une pile à un autre niveau d'une autre pile ; on n'en a pas forcément conscience en tant qu'utilisateur, ces passages sont transparents pour l'utilisateur de base, (en mode "survol"), mais on est obligé de naviguer dans ce "feuilleté" de niveaux dès qu'on veut organiser soi-même un "parcours" dans un stock d'informations (ce qu'on appelait programmer mais c'est différent sur HC).

.M1.ANNEXE T-4

SOURCE	Dans cette annexe, nous citerons très largement le texte de présentation générale des systèmes-experts, rédigé par Ph. Larvet, auteur du logiciel « Turbo-SE » et de l'ouvrage SYSTEMES EXPERTS EN TURBO-PASCAL, Paris, Editions EYROLLES, 1987.

COMMENTAIRE	Quelques précisions sur les systèmes-experts.

.M2.Les systèmes-experts

Un système expert (nous écrirons "SE" pour simplifier) est un logiciel destiné à assister l'homme dans un domaine où est reconnue une expertise humaine insuffisamment structurée pour constituer une méthode de travail directement transposable sur ordinateur.
Un SE tente de reproduire les facultés de décision ou de jugement des experts humains, en modélisant le comportement d'un spécialiste dans un domaine de connaissance précis.
Il **assiste** l'homme. Il n'est pas à même de le remplacer.
Etant donné que l'homme est, lui, capable de manipuler une énorme quantité d'informations, il n'est possible aujourd'hui de simuler sur ordinateur que des raisonnements dans des domaines très pointus, d'où le nom d' «experts» donné à ces systèmes.

.M3.Un système expert pour quoi faire ?

Les domaines d'application de la technique des SE sont aujourd'hui les suivants :

- Analyse de situation : expertise bancaire, financière, juridique, économique;
- Diagnostic : détection, diagnostic de panne, diagnostic médical, pré-diagnostic, maintenance;
- Aide à la décision : identification, agrément de matériel ou de configuration de systèmes divers, classification;
- Procédures : aide à la mise en place et au suivi de procédures industrielles, administratives, comptables, organisationnelles, méthodologiques, etc.
- Process control : contrôle "intelligent" de processus industriels;
- Formation : EAO, formation de nouveaux experts.

Les SE ne se limitent pas à des domaines complexes.
Ils peuvent, par exemple, rendre de grands services dans la résolution de problèmes simples de classification, dans un environnement bureautique.
Ils peuvent apporter une aide lors du suivi d'un projet, soit par la possibilité qu'ils offrent de simuler aisément un enchaînement de tâches et

d'en tirer les conclusions, soit en gardant la "mémoire" du projet et de l'emboîtement logique des étapes et phases qui le composent.
Ils peuvent aider à la recherche d'informations précises dans une grande masse de connaissances, par la possibilité qu'ils offrent de croiser aisément de nombreux critères de choix. Ils représentent ainsi un mode d'accès plus souple et plus naturel en environnement "bases de données".

.M3.Fonctionnement d'un système expert

Un SE fonctionne à partir de *connaissances* . Celles-ci sont exprimées simplement dans le système sous la forme de *règles* simples, écrites en français, dont le format général est

si (conditions) alors (conséquence).

Ces règles, c'est vous-même qui les écrivez, en tant qu'expert de votre domaine.
Selon la manière dont vous les avez libellées, selon l'enchaînement de ces règles, vous rendrez votre système plus ou moins "intelligent". Mais en tout état de cause, cette intelligence, c'est vous qui l'aurez introduite dans le système. Le système ne fera que reproduire les enchaînements et les déductions que *vous* aurez prévus.
Une autre manière de voir les choses est de considérer que vous disposez, avec un système expert, d'un outil qui vous permet de *programmer* vous-même facilement vos applications, à l'aide d'une seule forme d'expression (la règle) et dans le langage qui vous est le plus familier (le français courant).
Prenez exemple sur les Bases de Règles qui vous sont proposées sur la disquette et qui vous montreront, à travers quelques cas très rudimentaires, comment libeller, exprimer et enchaîner les règles dans une Base de connaissances.

.M3.Les éléments d'un système expert

Un SE se compose de trois éléments essentiels, que l'on retrouve dans TURBO-SE :

- une Base de Règles, qui contient les "connaissances" brutes du système (et qui pour cette raison est également appelée "Base de connaissances");
- une Base de Faits, qui contient les données de départ sur lesquelles le système commence à travailler, et qui s'enrichit au fur et à mesure des déductions faites par le système;
- un Moteur d'Inférence, qui est le coeur même du système, et qui contient un ensemble d'algorithmes permettant la déduction de nouveaux faits à partir des Règles (chaînage avant) ou, à l'inverse, la recherche des faits permettant de vérifier une hypothèse ou une conséquence données (chaînage arrière).

.M3.LES REGLES

TURBO-SE travaille à partir de Règles dites "règles de production" de la forme

si (ensemble de prémisses ou conditions)

alors (conséquence)
exprimées en français courant, et dans lesquelles les *prémisses* et la *conséquence* sont constitués de *faits*.

Par exemple :
SI la lampe est OK
ET le courant passe
ALORS la lampe s'allume

Les prémisses représentent les faits-conditions qui permettent la déduction du fait-conséquence si toutes les prémisses sont *vérifiées*.
D'une manière générale, on dira qu'une prémisse est vérifiée si le fait qui lui correspond est considéré comme *vrai* par l'utilisateur du système ou s'il a été déduit par le système lui-même.

.M3.LE CHAINAGE AVANT

En chaînage avant, le moteur d'inférence "raisonne" de manière déductive. Il *déduit* les faits-conséquences des Règles à partir des faits élémentaires entrés par l'utilisateur dans la Base de Faits.
Pour effectuer ces déductions, le moteur examine une à une les Règles de la Base de Règles et recherche chacune de ses prémisses dans la Base de Faits. La conséquence d'une Règle donnée ne peut être déduite (c'est-à-dire *vérifiée*) que si toutes les prémisses de la Règle sont présentes dans la Base de Faits. La règle correspondante est alors *déclenchée* .
La conséquence déduite d'une Règle vient ensuite enrichir la Base de Faits. Puis le processus de déduction se répète sur la nouvelle Base de Faits enrichie, tant que le moteur est capable d'effectuer de nouvelles déductions. Dès qu'un tour complet d'examen des Règles n'a entraîné la déduction d'aucune nouvelle conséquence, le fonctionnement du processus de chaînage avant s'arrête.
Illustrons ce fonctionnement par un exemple simple. Considérons les deux règles suivantes :

1 - si la lampe est neuve
alors la lampe est OK
2 - si le courant passe
et la lampe est OK
alors la lampe s'allume
L'utilisateur entre dans la Base de Faits les faits suivants :
le courant passe
la lampe est neuve

Au premier tour de déduction, le système peut déduire le nouveau fait :
la lampe est OK
qui vient s'ajouter, dans la Base de Faits, aux deux faits précédents.
Le second tour de déduction s'effectue alors sur une nouvelle Base de Faits contenant ce fait supplémentaire et permet une nouvelle déduction :
la lampe s'allume
Le moteur effectue ensuite un troisième tour de déduction, mais qui ne permet aucune déduction supplémentaire. Le processus de chaînage avant s'arrête alors.

.M3.LE CHAINAGE ARRIERE

Le chaînage arrière est utilisé pour permettre à l'utilisateur de connaître les faits qui, s'ils étaient exacts, seraient à même de vérifier une conséquence donnée. Autrement dit, ce processus permet de connaître les prémisses des Règles qui aboutiraient, après déduction, à la conséquence considérée.
Dans ce mode, le moteur "raisonne" donc en *remontant* d'un fait final vers les prémisses qui permettent de le vérifier. C'est pourquoi ce mode est également appelé *vérification* .
Le processus de chaînage arrière réalise ainsi une démarche d'*induction* vers les prémisses de plusieurs Règles, à partir de la conséquence (on peut aussi parler d'*hypothèse*) que l'on cherche à vérifier.
Cette recherche inductive se fait, comme dans le cas du chaînage avant, de manière récurrente : le processus inductif s'applique totalement aux prémisses que l'on trouve à chaque niveau, ce qui permet de remonter complètement vers la source de la déduction possible de la conséquence.

Prenons l'exemple des trois Règles suivantes :
1 - si (a)
alors (b)
2 - si (b) et (c)
alors (d)
3 - si (d)
alors (e)
Si l'on utilise le chaînage arrière pour chercher à vérifier la conséquence (e) - c'est-à-dire si l'on cherche à connaître les faits qui permettraient de vérifier l'hypothèse (e) - le système répondra qu'il faut et il suffit, pour que l'hypothèse (e) soit vérifiée, que les faits (a) et (c) soient vérifiés.

.M3.LE MODE EXPERTISE

Appelé "mode assisté" dans TURBO-SE, ce mode peut s'appliquer aussi bien au chaînage avant qu'au chaînage arrière.

Il propose à l'utilisateur de vérifier une à une les prémisses des Règles non encore vérifiées.
Dans le cas du chaînage avant, il est proposé automatiquement en cas de "non résolution", c'est-à-dire dès qu'aucune conséquence n'a pu être déduite des faits présents dans la Base de Faits.
Il est également proposé automatiquement dans le cas du chaînage arrière, pour chacune des prémisses induites à partir de la conséquence (ou de l'hypothèse) à vérifier.

Un système expert peut être construit selon différents principes. Celui dont vous avez fait l'acquisition est basé sur la notion de "système de production". Un tel système se définit comme étant un ensemble de "règles de production".
Une règle de production, du type

si A alors B

est une entité d'information que l'on peut écrire de manière formelle

A -> B

(que l'on lit "A produit B") et que l'on interprète comme suit :

si la situation A se manifeste alors on peut déclencher l'action B

cette interprétation revêtant une structure conditionnelle.
Cette structure met en évidence l'un des mécanismes fondamentaux de

fonctionnement d'un système de production, qui est le suivant :

de **{ p }**
et **{p => q}**
on conclut **{ q }**

que l'on interprète comme suit :

si p se réalise et si p => q alors q peut se réaliser

Ce schéma de calcul par règles de production a été proposé pour la première fois vers 1930 par le logicien américain Emil POST. Les systèmes formels qu'il inventa alors sont également appelés «systèmes de production de Post» [1]

Un système de ce type est un ensemble de *règles de transformation* de symboles opérant à partir d'un ensemble d'*axiomes* de départ. Le mécanisme de base de cette transformation est le schéma de calcul présenté ci-dessus.

Prenons un exemple : considérons les axiomes de départ

a = {D, H}

applicables sur le système de production suivant (le symbole "&" représente la conjonction "et") :

1 D -> C
2 D -> K
3 H **&** E -> I
4 A **&** I **&** K -> J
5 F **&** J -> A
6 B -> E
7 A -> E
8 C -> F
9 C -> B
10 J -> G

Le jeu consiste, en utilisant ces dix règles, à *produire* de manière automatique un ensemble de symboles *valides* . Les logiciens appellent ces symboles des *théorèmes* du système. Sur la base des axiomes et des règles ci-dessus, la question suivante, par exemple, peut être posée : «"J" est-il un théorème du système?» autrement dit : les règles du système permettent-elles de *produire* "J" à partir des axiomes de départ ?

A l'origine, l'ensemble des axiomes constitue l'ensemble des symboles valides, noté v :

v = a = {D, H}

A partir de cet ensemble, le mécanisme suivant est appliqué : si une règle possède, à gauche du symbole "->", des symboles qui appartiennent tous à v, alors les symboles de la partie droite de cette règle sont ajoutés à v, s'ils n'y sont pas déjà.

Faisons fonctionner ce mécanisme, à partir de l'ensemble v : au premier "tour" d'examen des règles, seules les règles 1, 2 et 3 peuvent être sélectionnées, ayant D ou H en partie gauche. Toutefois, E n'appartient pas à v, donc la règle 3 ne peut pas s'appliquer. Les règles 1 et 2, en revanche, s'appliquent et permettent de *produire* C et K : cette production constitue ce que l'on appelle une *inférence logique* .

Le nouvel ensemble v est à présent le suivant :

v = {D, H, C, K}

---

1 Philippe Larvet conseille la lecture du « *très remarquable ouvrage* » de Douglas HOFSTADTER, *Gödel, Escher, Bach*, InterEditions, Paris, 1985, page 38.

Continuons à faire fonctionner ce mécanisme, simple mais fondamental. La liste des v successifs est la suivante :

$v_0 = \{D, H\}$
$v_1 = v_0 \cup \{C, K\} = \{D, H, C, K\}$
$v_2 = v_1 \cup \{F, B\}$
$v_3 = v_2 \cup \{E\}$
$v_4 = v_3 \cup \{I\} = \{D, H, C, K, F, B, E, I\}$

Ce mécanisme représente la forme primitive du fonctionnement du *moteur d'inférence* d'un système-expert. Dans le cadre des systèmes formels de Post, il s'applique à des *symboles* , mais il est bien entendu applicable également à des données factuelles.

Prenons par exemple le cas d'un syllogisme simple

tout homme est bipède
or Paul est un homme
donc Paul est bipède

à partir duquel nous pouvons construire deux règles de production :

si homme alors bipède
si Paul alors homme

Il s'agit bien là d'un système de production, que l'on peut formaliser de la manière suivante :

homme -> bipède
Paul -> homme

Si l'axiome Paul est vérifié, les règles peuvent s'appliquer. Voici le mécanisme de cette inférence, directement inspiré de la démonstration ci-dessus :

1) Paul existe; c'est l'axiome de départ :
   v = {Paul}
2) A partir de cet axiome, le but du système est de chercher à produire des déductions, c'est-à-dire à déclencher les règles applicables.
3) Examen de la première règle : pas de déclenchement possible, car l'élément Paul ne figure pas dans la partie gauche de la règle.
4) Examen de la seconde règle : l'élément Paul figurant dans sa partie gauche, la règle est applicable et sa partie droite homme vient enrichir l'ensemble des éléments valides :
   v = {Paul, homme}
5) Second tour d'inférence et ré-examen de la première règle, qui permet cette fois de produire l'élément bipède, qui vient enrichir l'ensemble des éléments valides :
   v = {Paul, homme, bipède}

Le mécanisme d'inférence a ainsi permis la *déduction* de nouveaux éléments, que nous appelons des *faits* .

Dans un système expert (SE), ces nouveaux faits déduits, ainsi que les faits-axiomes de départ, sont rangés dans un espace de travail spécifique appelé *Base de Faits* . L'ensemble des règles de production du système sont stockées dans la *Base de Règles* . Enfin, le *moteur d'inférence* contient les algorithmes permettant la mise en oeuvre des mécanismes d'inférence. Le parallèle est donc clair entre système de production et système expert, ce dernier pouvant être considéré comme une extrapolation complexe du concept de système de production.

.M3.SOUPLESSE DES REGLES DE PRODUCTION

Malgré la dérivation que nous avons pu en faire ci-dessus, il faut noter qu'il existe une différence importante entre *syllogisme* et *règles de production* . Le syllogisme est construit et déduit de manière très rigoureuse, eu égard au fait qu'il représente un cas particulier de système de production très fermé, contenant une ou deux règles en plus de ses propres axiomes, celles-ci s'appliquant directement sur ceux-là.
Il n'en est pas de même du cas général des systèmes à base de règles de production.

Chacune des règles d'un tel système, en effet, doit être regardée autrement :

1) elle n'est vraie (c'est-à-dire qu'elle ne peut être validée, déclenchée) que si chacune de ses prémisses est vraie elle aussi;
2) elle peut être construite très librement : l'utilisateur a toute latitude pour choisir la structure des règles qu'il veut intégrer au système;
leur mode de construction et de déclenchement;
3) la déduction opérée à partir de ces règles est systématique, mais elle ne répond pas à un schéma pré-établi comme c'est le cas pour la résolution des syllogismes.

.M3.ORDRE ZERO et ORDRE 1

"TURBO-SE pour Mac" contient un moteur d'inférence d'*ordre zéro* .
L'*ordre* d'un SE détermine la nature des informations que ses règles peuvent traiter. Les prémisses et les conclusions d'une règle peuvent être en effet :
- des propositions logiques;
- des "prédicats" avec variables;
- des clauses plus complexes contenant des fonctions de variables, etc.

"**Ordre 0**" **signifie que les règles du système ne prennent pas en compte les variables et qu'elles ne sont donc constituées que d'énoncés**, de *propositions* . C'est pourquoi on parlera de calcul (ou "logique") des propositions. Par exemple :

si Japon matière-grise riche
alors Japon pays puissant
si Arabie sous-sol riche
alors Arabie pays puissant

En revanche, **un système d'ordre 1 autorise l'utilisation de variables à l'intérieur des règles**. Celles-ci sont alors basées sur la logique dite "du premier ordre" qui prend en compte le traitement des "prédicats" (le langage PROLOG est un exemple d'utilisation de ce type de logique). Par exemple, les règles ci-dessus s'écriront :

si matière-grise(X, riche)
alors pays(X, puissant)
si sous-sol(X, riche)
alors pays(X, puissant)

Il existe des moteurs d'ordre supérieur, "1+" par exemple, qui permettent l'utilisation de prédicats variables dans les règles. Dans ce cas, les deux règles ci-dessus peuvent s'écrire en une seule :

si Y(X, riche)
alors pays(X, puissant)

"Y" pouvant prendre par la suite les valeurs "matière-grise", "sol", "sous-

sol", etc.

L'**ordre 2** permettrait la description de règles beaucoup plus complexes, dans lesquelles les prédicats peuvent avoir pour variables des fonctions de variables.

Cette notion d'ordre découle directement de la logique mathématique. Le calcul propositionnel est dit d'ordre zéro, alors que le calcul des prédicats relève d'un ordre supérieur - on parle généralement de la logique du premier ordre.

TURBO-SE se contente modestement du calcul propositionnel [1]…

.M3.COMPOSITION D'UN SE

D'une manière générale, un SE contient trois éléments principaux, que l'on retrouve dans TURBO-SE:

- une Base de Règles (BR) ou Base de connaissances, qui contient la *connaissance* du système, exprimée sous forme de règles de production;
- une Base de Faits (BF), qui contient tout d'abord les axiomes ou faits initiaux (il s'agit, nous l'avons vu, des données de départ à partir desquelles le SE va commencer à fonctionner) puis qui s'enrichit au fur et à mesure des déductions du système;
- un Moteur d'inférence, qui est le coeur du système, et qui contient un algorithme de résolution dont la fonction consiste à effectuer des déductions en partant des **faits initiaux** et en s'appuyant sur les **règles** contenues dans la BR, dans le but final de produire (c'est-à-dire de déduire) de **nouveaux faits**.

.M3.FONCTIONNEMENT GENERAL DU SYSTEME

Le coeur du SE est bien entendu l'ensemble des algorithmes de résolution situés dans le moteur d'inférence. Le fonctionnement de ce moteur est fort simple. Explicitons-le à partir de la description des règles qu'il manipule. Celles-ci sont de la forme

si (ensemble de conditions) A
alors (conséquence) B

dans laquelle les deux ensembles (A) et (B) sont constitués de **faits.** L'ensemble (A) est constitué de faits-conditions qui sont les prémisses de la règle, et l'ensemble (B) ne contient qu'un seul élément : le fait-conséquence, qui sera déduit si tous les faits-conditions de l'ensemble (A) sont *vérifiés*. Un fait est considéré comme vérifié s'il existe dans la Base de Faits ou s'il a été déduit automatiquement par le système.
Afin de bien comprendre le fonctionnement même du système, il est intéressant de considérer tous les éléments constitutifs d'une règle comme des faits liées entre eux de manière logique. Dans notre SE, ces faits sont simplement énoncés sous forme de propositions logiques.

Considérons par exemple la règle suivante, extraite (et simplifiée) du

---

1 S'ils sont nettement moins puissants, les SE d'ordre zéro offrent au moins l'avantage de « faciliter » la mise au point des règles.

système expert MYCIN, destiné au diagnostic médical et principalement à l'analyse d'infections microbiennes :

si (l'infection est une primary-bacteremia)
A1
et (le site de la culture est stérile)
A2
et (l'introduction de l'organisme est dûe au tractus gastro-intestinal)
A3
alors (l'organisme est un bactéroïde)
B
Le fait (B) est le fait-conséquence, les trois autres faits (A1), (A2) et (A3) sont les faits-conditions.
Chaque règle est donc formée de deux ensembles distincts, organisés comme une suite logique de propositions.

.M3.AU COEUR DU MOTEUR D'INFERENCE

Le but du moteur d'inférence est de produire de nouveaux faits, c'est-à-dire de déduire des conséquences à partir des faits initiaux présents au départ dans la Base de Faits.
Pour y parvenir, il va parcourir la Base de Faits, examiner les faits un par un et, les considérant comme des faits-conditions, va chercher dans la partie gauche de chaque règle (l'ensemble A contenant les prémisses) s'il y a correspondance (c'est-à-dire égalité) entre l'une des conditions-prémisses de la règle et le fait considéré.
S'il n'y a pas égalité, le moteur passe à la règle suivante.
S'il y a égalité, l'unification est alors possible entre la prémisse de la Règle et le fait considéré de la Base de Faits. Deux cas peuvent alors se présenter :

- Soit la condition trouvée est unique dans la règle et suffit donc pour déclencher la conséquence (partie droite de la Règle, ensemble B).
  Par exemple :
  si (a) alors (b)
  => il suffit que le fait (a) soit "vrai", c'est-à-dire qu'il existe dans la Base de Faits - pour que (b) soit vérifié.
  Dans ce cas, la Règle est déclenchée et la conséquence peut être déduite : le fait (b) devient vrai à son tour. L'*inférence logique* réalisée permet au fait-conséquence de devenir un nouveau fait, qui est affiché à l'écran et vient enrichir la Base de Faits.
  Soit la condition trouvée n'est pas suffisante (elle n'est pas seule dans la partie gauche de la Règle) pour permettre le déclenchement de la conséquence.
  Par exemple :
  si (a) et (c) alors (b)
  => le fait (a) ne suffit pas pour déclencher la règle.

Dans ce cas, le moteur continue à explorer la Base de Faits à la recherche de faits correspondant aux autres faits-conditions exigés par la règle pour permettre son déclenchement. Puis le processus se répète.

Si l'ensemble des faits finalement trouvés dans la Base de Faits est suffisant, c'est-à-dire égal à l'ensemble A des prémisses de la Règle, celle-ci est déclenchée et son fait-conséquence est affiché et vient enrichir la BF; sinon le moteur passe à la règle suivante.

Le moteur d'inférence explore ainsi toutes les règles contenues dans la Base de Règles, cherchant pour chacune d'elles à déclencher sa conséquence.
A la fin de cette exploration, si un nouveau fait a été déduit, une nouvelle exploration est effectuée pour tenter de déclencher de nouvelles règles, et ainsi de suite jusqu'à "épuisement" de la Base de Faits.
Lorsque la BF a été ainsi examinée totalement et qu'aucun nouveau fait n'a pu être déduit à la suite d'une exploration complète de la Base de Règles, le moteur s'arrête.

L'ensemble du processus qui vient d'être décrit est celui du **chaînage avant**, c'est-à-dire du procédé permettant de *déduire* la conséquence d'une règle à partir de ses prémisses.
Le processus inductif de **chaînage arrière** correspond au mode de raisonnement inverse. Celui-ci, à partir du fait situé dans la partie droite d'une règle - et donc considéré comme une hypothèse de départ, ou une conséquence à vérifier - permet au système de *remonter* la chaîne logique des déductions possibles vers les faits initiaux qui permettraient le déclenchement des règles correspondantes et, de fil en aiguille, la vérification de l'hypothèse.

.M1.ANNEXE T-5

SOURCE	Documentations diverses. Recherches personnelles. Cf. bibliographie.

COMMENTAIRE	L'histoire de l'intégration des composants électroniques.

.M2.L'intégration des composants électroniques - Les familles de microprocesseurs

.M3.L'intégration : un principe simple.

Pour intégrer un circuit, on le dessine en grandeur nature, puis on réduit photographiquement les circuits en les projetant sur une plaque de silicium (*) rendu photo-sensible. Il « suffit » de faire en sorte que les différents composants (transistors, diodes, résistances, condensateurs) ne se confondent pas les uns avec les autres. On y parvient grâce à toute une série de traitements physico-chimiques.
Intégrer consiste donc à réduire la taille d'un circuit tout en lui conservant ses qualités, voire même en en accroissant certaines : si les distances sont plus courtes, les vitesses de calcul seront plus grandes...
Les premiers CI (Circuits Intégrés) contenaient environ un millier de composants sur une « puce » de silicium, enrobée d'isolant, mais très vite, de gros progrès d'intégration furent effectués avec la mise au point des circuits à haute intégration (10 000 composants sur 10 millimètres carrés, LSI en anglais), puis à très haute intégration (500 000 à 1 million de composants) et enfin à ultra-haute intégration (1 à 4 million de composants, VLSI)...

.M3.Les familles de microprocesseurs

Les microprocesseurs « 8 bits »

On parle indistinctement des micro-ordinateurs et des microprocesseurs à 8 bits, en ce sens que le seconds constituent le cœur (ou l'unité centrale) des premiers.
Ils ne sont capables **de traiter que 8 informations binaires à la fois**, c'est à dire 2^8 combinaisons, ou encore des nombres ne dépassant pas la limite de 256-1, soit 255 (en binaire, le nombre 255 se code 11 111 111, c'est à dire 100 000 000 - 1...). En fait on doit compter un bit de plus destiné à vérifier la parité, ce qui permet à la machine de s'y retrouver dans les suites ininterrompues de zéros et de uns qui ne cessent pas de défiler...
Au delà, le processeur doit « découper »les informations en plusieurs morceaux dont il s'occupera successivement, ce qui impose des vitesses de calcul (ou de traitement s'il s'agit par exemple de recherche ou de tris) qui pourront paraître assez lentes...

Ce type de micro-processeur est encore aujourd'hui fort répandu, notamment dans la gamme des matériels au standard « amateur »...

Les microprocesseurs « 16 bits »

Les 16 bits se caractérisent essentiellement par une capacité de traitement doublée puisqu'ils sont capables **de traiter simultanément des « mots » de 16 caractères** (c'est à dire des séquences de 16 « O » ou « 1 »)... Avec eux, il devient possible de concevoir des progiciels professionnels de forte puissance, principalement pour tout ce qui concerne la gestion de fichiers importants... A titre d'exemple, on peut dire aussi qu'ils permettent de traiter en une seule fois des nombres atteignant 2^{16}-1 soit 65 536 - 1... On en profitera pour constater l'énorme écart entre un 8 et un 16 bits, due au passage de 2^8 à 2^{16}.

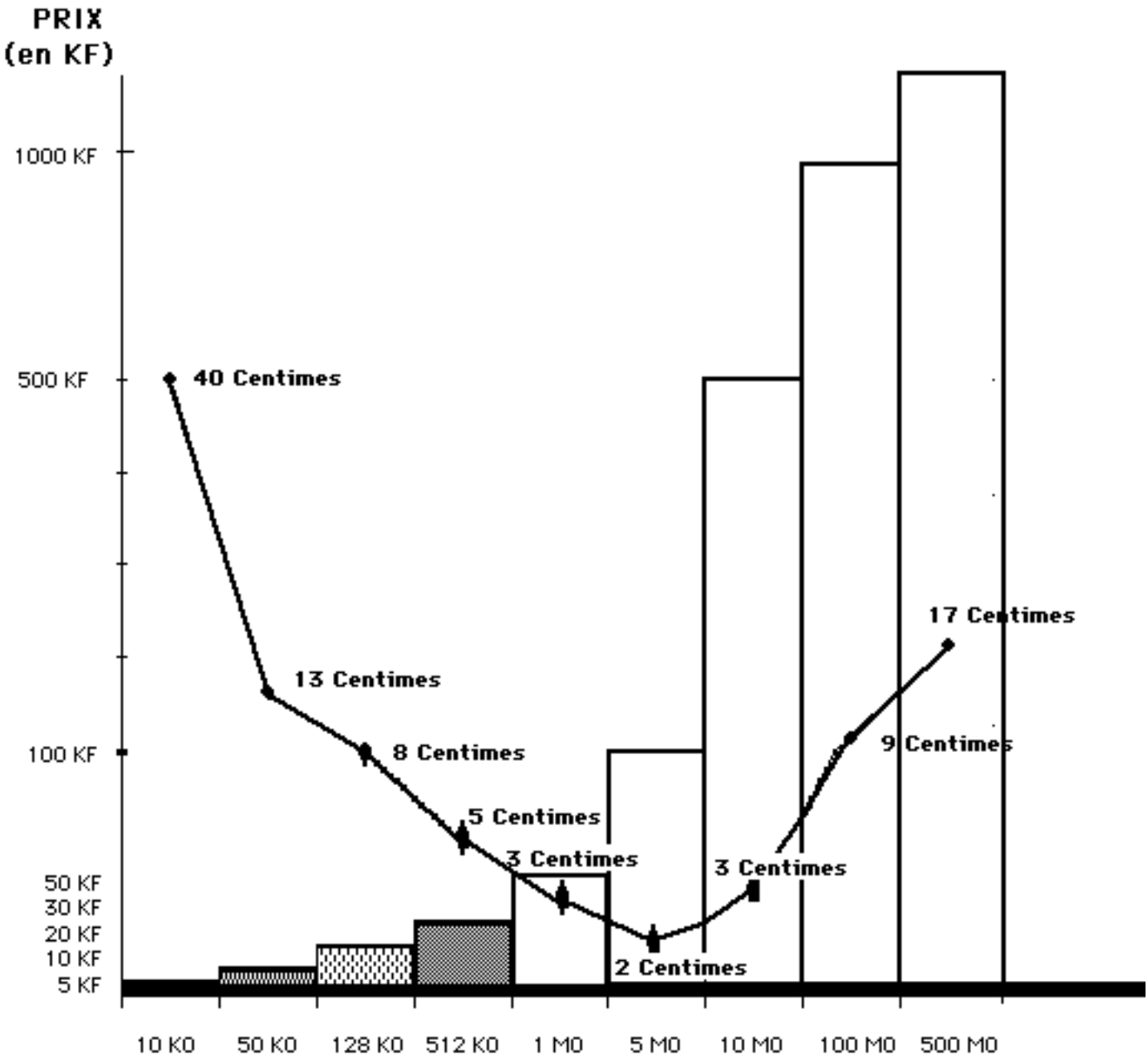
Les microprocesseurs « 32 bits »

Ils se caractérisent par une puissance encore accrue permettant de traiter **32 caractères en même temps.** En d'autre termes, on peut dire qu'ils sont capables de calculer directement jusqu'à 2^{32} , c'est à dire jusqu'au nombre 4 294 967 296 - 1 !...
Les 32 bits sont les processeurs de la fin de la décennie quatre-vingts, ils permettent d'effectuer rapidement des calculs complexes, mais le fait le plus important est qu'ils facilitent l'apparition de logiciels faciles à utiliser par des non-spécialistes, ce qu'en jargon informatique, on nomme des progiciels « *conviviaux* » …

.M3.La chute des coûts

Le prix des composants de micro-électronique ne cesse de baisser. On en trouvera un état comparatif sur la page suivante :

CAPACITES DE MEMOIRE CENTRALE (RAM ou MEV) ET PRIX DES UNITES COMPLETES (des micro aux méga-ordinateurs).



Nous indiquons en gras le prix moyen de l'octet de mémoire vive (RAM ou MEV).

.M1.ANNEXE T-6

SOURCE	Documentations diverses. Recherches personnelles. Cf. bibliographie.

COMMENTAIRE	L'histoire de l'intégration et de la fabrication des composants électroniques.

.M2.La fabrication des microprocesseurs

Nous n'entrerons pas dans le détail des méthodes de fabrication, lesquelles sont d'ailleurs gardées comme des secrets d'Etat et font l'objet de belles opérations d'espionnage industriel (comme l'affaire « Hitachi » aux USA). Nous nous contenterons d'en indiquer quelques-uns des traits les plus marquants.
La conception d'un nouveau micro-processeur occupe des équipes d'ingénieurs pendant plusieurs années. La mise au point est effectuée à l'aide d'autres ordinateurs très puissants, et la fabrication intégralement placée sous le contrôle de calculateurs spécialisés, considérés aux USA comme éminemment stratégiques. C'est d'ailleurs là un des gros problèmes de l'industrie française du micro-processeur, basée essentiellement à Grenoble. Les entreprises qui voudraient se spécialiser dans ce genre de production ne peuvent avoir qu'au compte-goutte les ordinateurs américains dont elles auraient besoin).
La fabrication des micro-processeurs s'effectue encore pour la plus grosse partie dans la célèbre « Silicon valley » en Californie. Mais des concurrences, entre autres japonaises s'annoncent, et la « Valley » va devoir dépenser beaucoup d'argent, d'efforts et d'énergie pour garder sa position mondiale dominante.

Un des traits les plus étonnants de la fabrication des micro-processeurs réside sans doute dans les « contrôles de qualité ». Ils sont comme on s'en doute extrêmement sévères et il suffit qu'une puce sur mille soit déclarée « hors normes » pour que les 999 autres, fabriquées en même temps se retrouvent au pilon (c'est à dire à la refonte), ou soient carrément jetées (le matériau ne coûte pas cher).
Mais le plus étonnant est ailleurs :
Etant donnée la complexité folle d'un microprocesseur à très haute, ou à ultra haute intégration, on ne peut tester convenablement toutes les fonctions, de sorte que les fabricants savent bien que certains circuits ont de fortes chances, quelque part dans la puce de ne pas fonctionner correctement... Ce qui n'a pas beaucoup d'importance, car d'autres circuits « ailleurs » vont forcément « doubler » les défectueux, de sorte que le fonctionnement global apparaîtra correct (les erreurs seront annulées).

Les concepteurs en tiennent compte et prévoient des taux de recouvrement suffisamment larges pour se garantir contre une trop forte proportions de circuits défectueux...

.M1.ANNEXE T-7

SOURCE	Documentations diverses. Recherches personnelles. Cf. bibliographie. Extrait de *L'informatique et nous*, op. cit., p. 223, sqq.

COMMENTAIRE	Quelques mots sur les tableurs.

.M2.Les tableurs

Les tableurs se partagent avec les systèmes de traitement de texte l'honneur d'avoir été dès 1982 (il y a déjà si longtemps...) les premiers progiciels standards offerts en «prêt-à-l'emploi» aux propriétaires de micro-ordinateurs qui en avaient plus qu'assez assez de programmer pendant des heures des applications qui auraient dû être simples et faciles à réaliser...

Pour bien faire saisir ceci, nous prendrons un bref exemple : celui d'un **programme de feuille de paie** à établir sur micro-ordinateur, par exemple en BASIC (lequel langage se révèle bien suffisant pour une question aussi facile).

Nous supposerons que l'analyse du problème soit faite (en fait, cette application est si simple qu'il n'y a sûrement pas d'analyse proprement dite à effectuer !...). On peut donc programmer tout de suite les variables à « entrer » (p.e. le salaire brut), les formules de calcul des retenues et autres indemnités (c'est facile puisqu'il ne s'agit que de pourcentages), puis les variables à «sortir» c'est à dire à afficher ou à imprimer...

Malheureusement c'est là que tout se complique, car s'il demeure extrêmement facile d'obtenir les réponses souhaitées (une feuille de paie juste...), il faut parfois passer un temps énorme à «soigner les affichages, ne serait-ce que pour obtenir des colonnes régulières, bien alignées et bien disposées sur l'écran et sur l'imprimante. (au moins aussi bien que dans une feuille de paie classique élaborée avec du papier et un crayon ou une calculette...). Ceci tient au fait que les langages informatiques comme le BASIC (et pratiquement tous les autres, sauf pour ce genre de problème le COBOL qui souffre par ailleurs de quelques autres défauts) ne sont pas du tout adaptés à cet usage. Ce qui impose de chercher des solutions pour le moins acrobatiques.

En résumé, pour une application aussi simple, on doit ruser avec le système et avec le langage, ce qui peut sembler amusant si l'on n'a vraiment rien d'autre à faire, mais risque de devenir bien vite un vice rédhibitoire.

C'est dans ce contexte que naquirent les tableurs, avec lesquels notre petit problème de feuille de paie peut être traité en quelques minutes avec à la sortie un affichage impeccable sur l'écran ou sur l'imprimante.

Naturellement, les tableurs sont capables d'autres performances, et ils ont été inventés pour répondre à des besoins multiples, mais si nous avons

relaté cet exemple, c'est pour mieux situer leur extrême importance dans le développement d'une informatique plus proche de ses utilisateurs, plus facile à saisir ou à appréhender parce que bientôt débarrassée de ses "pêchés de jeunesse". *Ajoutons encore que les tableurs ont marqué la double apparition sans doute irréversible des progiciels paramétrables et des programmes d'aide à la décision...*

.M3.Ce qu'est un tableur...

Les tableurs, parfois appelés aussi « feuilles de calcul électronique » commencent à être bien connus, mais il n'est peut-être pas inutile de rappeler leur principe élémentaire : Un tableur se présente sous la forme d'une feuille distribuée en lignes et en colonnes dont on peut observer une (petite) partie sur l'écran. Des commandes permettent d'explorer toute la feuille en balayant ses lignes et ses colonnes.

Pour utiliser le tableur, il suffit de définir les relations que l'on veut créer entre les lignes et les colonnes, par exemple un calcul de retenues sociales à partir d'un salaire brut.

Là où l'on mesure toute la puissance du système, c'est lorsque l'on entre des données dans les cases d'entrée et que l'on voit les cases de sortie se remplir ou se modifier instantanément sans qu'il soit besoin de donner la moindre instruction. Qui plus est, les résultats sont toujours bien présentés et alignés, ceci sans le moindre effort.

Il est facile d'imaginer les possibilités infinies de tels logiciels, et ce n'est pas un hasard si le premier d'entre eux VISICALC a été vendu, malgré un sévère piratage à plus de 500 000 exemplaires... Le leader des années 1980-1984, MULTIPLAN en est déjà à un million lui aussi... On peut citer « 123 » ou Excel qui rivalisent de puissance (256 colonnes sur 16384 lignes pour ce dernier, soit 4 194 304 cases !!).

LES PRINCIPALES FONCTIONS D'UN TABLEUR

Les fonctions apparaissent si nombreuses et variées que nous ne citerons que les principales :

1. Le calcul et le recalcul automatique des cases : cette fonction est naturellement la plus connue, précisons que selon la puissance des ordinateurs utilisés, on peut atteindre des tableaux de plusieurs centaines de colonnes et autant de lignes, ce qui représente des dizaines de milliers de cases à recalculer à chaque nouvelle introduction de données...
2. Les répétitions automatiques de formules : il suffit de taper une formule et d'indiquer au système les cases entre lesquelles il doit la recopier (éventuellement en l'adaptant).
3. Les changements de largeurs de colonnes et de longueurs de lignes : les premiers tableurs (VISICALC de la première génération) imposaient des colonnes de largeur fixe, ce qui n'était pas toujours très pratique dans certaines applications.
4. Les possibilités de comparaisons et de tris de données : les tableurs de la dernière génération permettent de traiter simultanément des chiffres et des lettres (ou plutôt des données numériques et alphanumériques), ce qui se révèle extrêmement pratique à chaque fois que l'on demande au programme de décider lui-même des suites à donner à une opération quelconque (on trouve des instructions du genre «si le chiffre d'affaire de telle branche d'activité atteint telle limite, alors reverser sur tel compte, tel pourcentage pris sur tel autre compte).

5. Les échanges d'informations avec d'autres logiciels : ainsi que nous l'avons vu au sujet des progiciels de traitement de texte et de gestion de bases de données, il apparaît indispensable que tous ces programmes puissent communiquer les uns avec les autres, ce qui a commencé à se réaliser seulement en 1983/84. On trouve même à présent des progiciels spécialisés dans ce genre de tâches. Malheureusement, il ne s'agit souvent que de "raccommodages" pas toujours faciles à exploiter...

LES TABLEURS DE L'AVENIR...

Il semble risqué de tenter de cerner la physionomie des tableurs des prochaines années. Ceux-ci ayant apparemment atteint une sorte de point d'équilibre du côté des applications purement numériques, ce sera certainement vers le traitement des données alphanumériques et graphiques que les feuilles de calcul ont le plus de chance de se tourner, à condition que les possibilités d'intégration grandissante des progiciels standards se poursuivent sur la pente actuelle. Avec ces tableurs, il deviendrait possible de simuler des décisions à partir de facteurs numériques et non numériques, un peu comme le faisait (toutes proportions gardées) les gros ordinateurs de la RAND CORPORATION au moment de la crise de CUBA . En d'autres termes, il deviendra peut-être possible d'évaluer les conséquences psychologiques, économiques, sociales ou politiques de décisions «microscopiques» parce qu'opposées aux « macroscopiques » que nous côtoyons tous les jours .

Naturellement, tout le réel ne pouvant se quantifier (on peut au moins l'espérer...) il y a peu de risque que l'on aboutisse irrémédiablement à une société dans laquelle toutes les décisions seraient soumises au choix des machines...

Nous retrouverons donc nos grandes catégories des principaux progrès à attendre. Elles tourneront toujours autour d'une plus grande « *convivialité* », d'une meilleure intégration et d'une plus puissante interconnexion avec d'autres données ou d'autres logiciels, et enfin d'une assistance « intelligente » du tableur lui-même pour que ses utilisateurs parviennent à en tirer le maximum de renseignements.

.M1.ANNEXE T-8

SOURCE	Documentations diverses. Recherches personnelles. Cf. bibliographie. Extrait de *L'informatique et nous*, op. cit., p. 269, sqq.

COMMENTAIRE	Quelques mots sur les langages informatiques.

.M2.Les principaux langages informatiques et leurs usages

Il existe *plusieurs centaines* de langages informatiques, chacun étant spécialisé dans un domaine bien déterminé d'applications (gestion financière, gestion de graphiques ou de sons, calcul scientifique, traitement de fichiers, intelligence artificielle, etc.).

Cette accumulation suffirait à elle seule à montrer que s'il en existe tant, c'est qu'aucun ne peut prétendre à être universel. Il ne faudrait pas oublier d'autre part que la plupart de ces langages n'ont pas été élaborés pour *rendre service* à un futur utilisateur mais plutôt pour correspondre à des besoins que leurs concepteurs ont imaginé pour lui. Pour aller plus loin, nous dirions même que la quasi-totalité des langages ne sont que des manifestations de tour de force (on peut dire aussi des thèses universitaires...) face aux contraintes multiples imposées par les machines, sans autre préoccupations que celles de résoudre à tout prix son problème sans s'occuper de ceux du voisin (!) qui programmera plus tard... En d'autres termes, on commence par essayer de "réaliser quelque chose de nouveau" (de préférence qui "fasse bien") et chemin faisant, on en crée un nouveau langage informatique (il existe heureusement des exceptions)...

Les **langages naturels** qui nous permettront de nous faire comprendre d'un ordinateur sans devoir passer par l'apprentissage d'un quelconque idiome spécialisé sont encore loin de nous, et en attendant, force nous est faite de les apprendre, sans jamais perdre de vue qu'il ne s'agit que de "***béquilles pour la pensée*** ", et pas encore de véritables instruments créatifs.

.M3.Tableau récapitulatif :

LANGAGE	Date de sortie	MICRO O/N	SIGLE OU ACRONYME	USAGES PRINCIPAUX
ADA	79	N	Inspiré d'ADA de Lovelace, collaboratrice de C Babbage (cf. page **)	Le langage de la décennie 80... Très structuré (un peu comme PASCAL) Usages généraux sur des applications complexes...
ALGOL	58	N	**ALGO**rithmic **Language**	Largement dépassé... Avait l'ambition d'être "universel"
APL	65	N	**A P**rogramming **Language**	Spécialisé dans la gestion de grands tableaux alphanumériques et les manipulations afférentes.
BASIC	65	O	**B**eginner's **A**ll purpose **S**ymbolic **I**nstructions **C**ode.	LE langage universel, excellen nulle part, mais capable de presque tout, de la gestion des sons, ou du graphisme jusqu'à celle des fichiers ou il n'excelle pas particulièrement.
BASIC FRANCAIS	80	0	Mentionné pour mémoire, car on l'oublie trop souvent, ce qui est dommage…	
LANGAGE C	78	O	Un langage qui s'accorde très bien avec les processeurs 32 bits et le système UNIX.	Très puissant pour les gestions de fichiers complexes. Avec C, il faut tout réinventer. En contrepartie, n'importe quelle fonction peut être appelée par n'importe quelle autre
COBOL	61	O	**CO**mmon zusiness **O**riented **L**anguage	Spécialisé dans la gestion financière Il est lourd, verbeux et sûrement "increvable" malgré ses défauts (peut-être parce que c'est encore l le plus parlé par les informaticiens de la grande informatique...
COMAL	80	O*	Savant mélange de **BASIC** et de **PASCAL**...	Tentative (peu connue) de réunir deux des meilleurs langages actuels en prenant à chacun ce qu'il a de meilleur. Le résultat n'est pas tout à fait à la hauteur des espérances…
FORTRAN	57	0	**FOR**mula **TRAN**slating system	Le premier de tous les langages informatiques. Malgré son âge, il est encore capable de rendre bien des services pour des calculs scientifiques, usage pour lequel il a été conçu.
FORTH	73	O	Signifie modestement (!) "PUISSANT"	Il est d'un emploi déroutant, mais effectivement très "puissant" dans la gestion de fichiers à organisation complexe avec de multiples sous niveaux. A n'utiliser que si l'on est sûr qu'aucun autre ne fera l'affaire
LISP	60	O	**LIS**t **P**rocessing	Il a inspiré ADA et est spécialisé dans l'intelligence artificielle et dans une moindre mesure les applications de robotique. Abord déroutant.
LOGO	71	O	Comme "esprit"... Un ouvrage le décrivant (cf. bibliographie) s'intitule "des ailes pour l'esprit" longtemps cantoné dans des applications du genre "dessine moi un carré"...	Développé pour l'initiation à la démarche informatique de jeunes enfants à partir des thèses de Jean **PIAGET**. Procédural et structuré, il est puissant. On l'a trop ongtemps

LSE	72	O *	**L**angage **S**ymbolique d'**E**nseignement pour l'enseignement, on ne le trouve que sue les matériels "Education Nationale". Il est **procédural** et **structuré** et s'affranchit assez bien des chaînes de caractères. Il est plus lourd pour les fichiers. Son avenir ne semble pas très rose malgré de grandes qualités...	Développé comme son nom l'indique
PASCAL	69	0	En hommage à Blaise PASCAL	Le premier langage vraiment structuré et semi procédural. Excellentes vertus pédagogiques malgré quelques abords déroutants au tout début. Très grandes qualité dans la gestion des fichiers indexés... Sans conteste un des meilleurs langages actuels (surtout dans ses versions les plus récentes).
PL/1	64	N	**P**rogramming **L**anguage des parenthèses rappelera de bons souvenirs de nuits blanches aux imprudents qui s'y seraient essayé...	Complètement dépassé... La gestion
HYPERCARD	87	O	Langage « orienté objet »	Le premier langage objet développé pour le grand public (cf. paragraphe spécial).

LEQUEL CHOISIR ?

Comme on s'en doutera, il est presque impossible de répondre à une telle question de manière complète et satisfaisante pour tous les cas...

Cependant, si l'on ne peut répondre à la question "lequel choisir ?", on peut néanmoins indiquer ceux qui nous semblent les meilleurs pour un usage "tout venant" sans domaine bien particulier :

1. Initiation de jeunes enfants : **LOGO** (à la rigueur BASIC)
2. Initiation à la programmation : **PASCAL**
3. Quand on ne peut pas faire autrement : **BASIC**
4. Pour les applications graphiques et l'exploitation des liens : **HYPERCARD**