LE FACTEUR HUMAIN A SONNÉ TROIS FOIS

Trois livres sur l'ergonomie des systèmes informatiques

Bertrand Meyer

- Ben Shneiderman : *Software Psychology : Human Factors in Computer and Information systems*, Winthrop (Cambridge, Massachusetts), 1980, ISBN 0-87626-816-5.
- Henry Ledgard, Andrew Singer, John Whiteside : *Directions in Human Factors for Interactive Systems*, Springer-Verlag (Berlin, Heidelberg, New York), 1981, ISBN 3-540-10574-3 et 0-387-10574-3.
- Stuart P. Card, Thomas P. Moran, Allen Newell : *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates (Hillsdale, New Jersey), 1983 ; ISBN 0-89859-243-7.

Comment faut-il faire pour construire des systèmes informatiques agréables à utiliser ? Question naïve, certes, mais dont tout utilisateur des systèmes actuels reconnaîtra l'importance. Les spécialistes de ce domaine de recherche lui donnent des noms divers : ergonomie des systèmes informatiques, étude des facteurs humains, psychologie du logiciel, *human engineering*, psychologie des utilisateurs, science des utilisateurs (*user science*), génie de la connaissance (*cognitive engineering*), etc. L'absence d'un vocable unique est sans doute un signe de la jeunesse de cette discipline et de l'absence de résultats définitifs. Mais son développement actuel répond indéniablement à un besoin ressenti de plus en plus largement.

Les trois livres analysés ici abordent ce domaine selon des points de vue assez différents. Le plus ancien (1980) est celui de Shneiderman, qui couvre en fait un domaine plus large, s'intéressant à tous les problèmes psychologiques liés au logiciel, c'est-à-dire non seulement à l'utilisation des programmes (le point qui nous occupe ici) mais aussi bien à leur réalisation. L'ouvrage de Ledgard, Singer et Whiteside (1981) s'attache, lui, à l'étude des « facteurs humains » ; comme Shneiderman, les auteurs sont des informaticiens qui se sont demandé comment les systèmes qu'ils réalisent peuvent être rendus plus agréables d'abord. L'esprit de l'ouvrage le plus récent, celui de Card, Moran et Newell — tout frais paru au moment où j'écris cette analyse — est différent : il s'agit d'une étude très serrée des aspects « cognitifs » de l'utilisation des systèmes interactifs, fondée sur l'utilisation des résultats actuels de la recherche en psychologie de la connaissance. L'approche est pluridisciplinaire ; si l'un au moins des auteurs (Newell) est bien connu dans le milieu informatique (et celui de l'intelligence artificielle en particulier), on pourra noter que la plupart des références bibliographiques sont extraites de revues telles que *Cognitive Psychology*, *Journal of Experimental Psychology*, etc.

1. Shneiderman

Comme indiqué plus haut, l'ouvrage de Shneiderman a un objet plus large que les deux autres. Le thème général est l'« approche psychologique » des problèmes de la programmation, et ceci recouvre des sujets aussi divers que la façon d'animer une équipe de programmeurs, le style de programmation, l'évaluation de la qualité du logiciel, les systèmes de gestion de bases de données, l'utilisation du langage naturel et la conception des interfaces interactives. Il n'est pas désagréable de trouver ainsi concentrées des réflexions sur des domaines aussi divers ; sur la plupart des questions traitées, Shneiderman est plus précis que [Weinberg 71], le premier ouvrage en date sur la psychologie de la programmation, qui apparaît aujourd'hui comme un peu simpliste. Mais une synthèse aussi vaste que celle de Shneiderman ne pouvait éviter une certaine dispersion.

Le sujet traité aurait dû, à mon sens, faire l'objet de deux livres distincts, correspondant à la division qui apparaît nettement dans l'ouvrage publié (chapitres 1 à 6 et 7 à 11 respectivement). Le premier sujet est celui des aspects psychologiques de la *réalisation* de logiciel ; le second, l'étude psychologique de *l'utilisation* des systèmes informatiques, et des conclusions que devraient en tirer les concepteurs.

La première partie constitue en quelque sorte un résumé de quelques unes des principales réflexions actuelles en génie logiciel, et traite aussi bien des structures de contrôle que de la « science du logiciel » ou des modèles de fiabilité. Il s'agit dans tous les cas de sujets sur lesquels il existe déjà une abondante littérature ; l'originalité de Shneiderman est l'éclairage psychologique apporté à tous ces sujets, et l'utilisation systématique de résultats quantitatifs, provenant en général de tests statistiques. Je ne suis pas pour ma part un fanatique des méthodes quantitatives dans ce domaine ; il me semble que les progrès significatifs en matière de méthodes et de langages de programmation ont été obtenus par une démarche plus déductive qu'inductive (opinion que l'on pourrait d'ailleurs contredire en rappelant que le fameux article de Dijkstra sur la nocivité des branchements commençait par la non moins fameuse remarque selon laquelle « [l'auteur] a remarqué depuis longtemps que la qualité des programmeurs était une fonction inverse du nombre de GOTO dans leurs programmes », ce qui est certainement un argument, sinon véritablement quantitatif, du moins inductif). Mais il n'est pas inintéressant, même si l'on n'a aucun doute, par exemple, sur la garantie de sécurité qu'offrent les langages fortement typés, de lire ou de relire les résultats de l'étude de Gannon [Gannon 77], reproduits par Shneiderman, qui appuient ce point de vue en se fondant sur des expériences systématiques.

La seconde partie, dont le sujet entre plus directement dans le cadre de cette revue comparative, souffre elle aussi d'une certaine dispersion. Deux chapitres traitent des bases de données, sujet auquel Shneiderman a précédemment consacré plusieurs publications. Était-il vraiment nécessaire d'inclure toute une discussion sur les différents modèles de données ? Je ne le pense pas ; la présentation que Shneiderman offre de ces questions en vaut une autre, mais elle appartient à un autre livre. En définitive, seules quelques pages, sur la cinquantaine consa-

crées aux bases de données, sont vraiment dans le sujet; ce sont celles qui discutent de la forme que doit revêtir un bon langage de requêtes.

Un chapitre est consacré à l'utilisation du langage naturel pour la communication avec les ordinateurs. Shneiderman fait preuve sur ce sujet (avec justesse me semble-t-il) d'une grande réserve, montrant qu'il n'y a aucune raison de postuler que le même type de formalisme est optimal pour la communication avec les êtres humains et avec les machines.

Les deux chapitres consacrés à la conception des systèmes interactifs et de leurs interfaces externes ont été pour moi les plus intéressants du livre. Le chapitre 11 en particulier (« Designing Interactive Systems ») présente les résultats de nombreuses études, dont beaucoup sont d'accès difficile par ailleurs. Il est instructif d'examiner et de comparer les recettes proposées par les différents auteurs pour la conception de systèmes interactifs de qualité; on est moins frappé par les désaccords qui apparaissent (bien que certains soient très nets) que par le caractère encore empirique de beaucoup de travaux en ce domaine, et par l'absence de critères généraux sur des problèmes tels que le traitement des erreurs ou la facilité d'apprentissage.

Certains conseils paraissent en fait contestables. Ainsi, de nombreux auteurs, et Shneiderman lui-même, affirment à la suite de Hansen [Hansen 71] que le premier principe est de bien connaître les futurs utilisateurs (*Know the user*). Suivi à la lettre, ce précepte me paraît dangereux. De nombreux systèmes — en fait, tous les systèmes qui réussissent — finissent par être utilisés par une communauté beaucoup plus large que celle pour laquelle ils ont été conçus initialement; ils pâtissent alors d'avoir été trop spécifiquement adaptés à leurs premiers utilisateurs. Deux exemples, appartenant à des domaines très différents, sont Fortran et Unix : dans les deux cas, la diffusion a dépassé de façon inespérée la cible initiale. Dans une situation de ce type, on constate que les systèmes imposent des conditions d'utilisation qui se justifiaient pour la première vague des utilisateurs mais n'ont plus aucune signification pour les nouveaux. On peut bien sûr répondre que, si de tels systèmes ont « réussi », c'est précisément qu'ils allaient comme un gant à leurs premiers utilisateurs. Mais il me paraît plus fructueux d'étudier les caractéristiques communes à tous les utilisateurs de systèmes informatiques que de se concentrer trop exclusivement sur les traits spécifiques de tel ou tel groupe qui, si le système est un succès, deviendra vite minoritaire.

Une partie de la discussion du chapitre 10, sur des points tels que les avantages respectifs de la sélection par menu ou par commandes, semblera bientôt, à mon sens, aussi dépassée que l'est dès aujourd'hui une autre partie de ce chapitre, celle qui se demande gravement s'il vaut mieux travailler en différé (*batch*) ou en interactif. Les progrès du matériel auront vite fait de trancher par l'exemple ce genre de débat. En outre, les utilisateurs eux-mêmes n'accepteront plus n'importe quoi : les enfants de l'ère des jeux vidéo seront certainement moins indulgents vis-à-vis d'éditeurs de textes à la syntaxe tortueuse que leurs parents, formés à la rude école du *batch* et des cartes perforées. Le livre de Shneiderman est assez sommaire quant aux nouvelles perspectives offertes par les moyens actuels de communication homme-machine; on le complétera utilement, sur ce point, par un article plus récent du même auteur [Shneiderman 83], consacré à ce qu'il appelle la *manipulation directe*. Dans cet article, Shneiderman examine un certain nombre de systèmes interactifs dont les utilisateurs sont satisfaits, voire enthousiastes. L'objet même de ces systèmes est varié : enseignement assisté par ordinateur, CAO, édition de textes « pleine page », jeux vidéo. Le trait commun que l'auteur a décelé dans ces différentes réalisations est qu'elles fonctionnent en manipulation directe : à chaque instant, l'utilisateur dispose d'une représentation explicite (et souvent graphique) de l'état actuel du système; chaque action qu'il effectue donne lieu à une modification immédiate de cette représentation. *On voit ce qu'on a; on voit ce qu'on fait.* Le sentiment de sécurité et de puissance qu'apporte ce mode de fonctionnement explique son succès; on peut penser que les systèmes interactifs de l'avenir se rattacheront de plus en plus systématiquement à ce schéma.

De même que les rames de métro ont toujours un dernier wagon (qu'un humoriste proposait de supprimer parce que c'est toujours le plus bondé), il y a toujours, dans un livre, un dernier chapitre. Dans le cas de celui de Shneiderman, on peut déplorer cette loi; je ne pense pas en effet que ses exhortations finales (« Computer power by and for the people ») apportent grand-chose. C'est plein de sentiments vertueux, mais appartient trop exclusivement à ce qu'on nomme en anglais (américain ?) « maternité et tarte aux pommes ». Le coup de grâce a été pour moi l'apparition d'une série de citations tirées du livre de R. Pirsig (*Le Zen et l'art de réparer les motos*) qui, très en vogue pendant un certain temps auprès d'une frange de la jeunesse intellectuelle américaine, est difficile à prendre au sérieux pour un lecteur européen. J'avoue que je me suis arrêté là dans la lecture de l'ouvrage de Shneiderman; mais il ne restait que quelques pages.

Ce livre, comme on l'aura vu, peut provoquer certaines irritations. Mais de la longueur même de cette analyse on pourra déduire qu'en dépit de toutes les réserves émises plus haut je le considère comme important et utile. Souhaitons que l'auteur nous donne quelque jour, sur un sujet mieux délimité, un ouvrage plus dense et plus profond.

2. Ledgard, Singer, Whiteside

Directions in Human Factors for Interactive Systems : le titre est alléchant. Malheureusement, ce livre souffre de plusieurs défauts. Tout d'abord, formé d'une série d'articles, il est un peu disparate. Mais, surtout, il est souvent superficiel et retarde, à mon sens, sur la technique.

Son caractère superficiel est particulièrement apparent lorsqu'on considère le chapitre sur « La spécification et la conception formelles ». Ce chapitre commence par un plaidoyer en faveur de l'utilisation de méthodes formelles; très bien, très bien. Et puis... c'est tout ! Le chapitre se termine là, après six pages. Quelles sont ces fameuses méthodes formelles, comment les applique-t-on, qui s'en est déjà servi ? Motus. Il n'y a même pas de renvoi bibliographique adéquat; seules deux références (l'une relative à VDL, l'autre à un langage pour l'EAO) se rapportent vraiment au sujet, mais on ne trouve rien sur VDM, par exemple, ni HDM-Special, FDM, etc.

Une partie importante du livre (près du tiers) est consacrée à la conception et à l'analyse des qualités d'un « assistant Pascal annoté », sorte d'éditeur de textes spécialisé. Il s'agit d'un outil de type « ligne à ligne », domaine d'étude qui me paraît peu adapté à la recherche avancée sur les « facteurs humains ». La thèse principale des auteurs est que l'interaction avec les systèmes informatiques doit utiliser autant que possible le langage

naturel. Shneiderman, on l'a vu, était plus réservé sur ce point. Dans le cas de l'« assistant Pascal », le résultat de la conception proposée est un système qui semble lourd et verbeux. Personne, à mon sens, ne préférerait un tel outil à un bon éditeur de texte « pleine page » ni aux systèmes de traitement de texte disponibles aujourd'hui sur micro-ordinateurs, tel Microsoft Word (cette rubrique est libre de toute publicité). L'« Assistant » n'est pas, en outre, particulièrement adapté à Pascal; ce n'est pas un « éditeur de structures » au sens de Mentor, Gandalf ou CPS, outils qui ne sont d'ailleurs pas cités.

A l'appui de leur thèse, les auteurs présentent des résultats de tests, aux cours desquels différents sujets ont utilisé, soit un éditeur de textes classique employant des conventions peu régulières pour la syntaxe des commandes, soit un autre éditeur assurant les mêmes fonctions mais avec une syntaxe plus proche de l'anglais (du type CHANGE ALL « BOY » TO « GARCON », etc.). Les tests montrent que l'utilisation du second est plus profitable sous tous les rapports. Mais ces résultats me semblent de portée assez limitée : les sujets ne connaissaient initialement aucun des deux éditeurs; or il est évident que, pour des novices, une syntaxe se rapprochant de celle de leur langue naturelle est plus facile à maîtriser. La comparaison ne donnerait pas nécessairement les mêmes résultats après six mois d'expérience. Mais surtout, je ne pense pas que la différence entre :

DELETE 8 (formulation requise par l'éditeur « classique »)
et
DELETE 8 lines (formulation requise par l'éditeur « anglais »)

soit vraiment fracassante. Il aurait été plus intéressant de comparer un éditeur ligne à ligne et un éditeur pleine page, de chercher si la sélection dans un menu vaut mieux que le recours à des commandes explicites, d'étudier les différents modes de pointage (curseur, souris, etc.), d'essayer de mesurer l'impact d'un environnement à plusieurs fenêtres en recouvrement (idée défendue avec enthousiasme par de nombreux constructeurs actuels d'environnements de programmation, mais dont l'efficacité réelle n'a pas à ma connaissance fait l'objet d'évaluations systématiques; peut-être un lecteur me démentira-t-il).

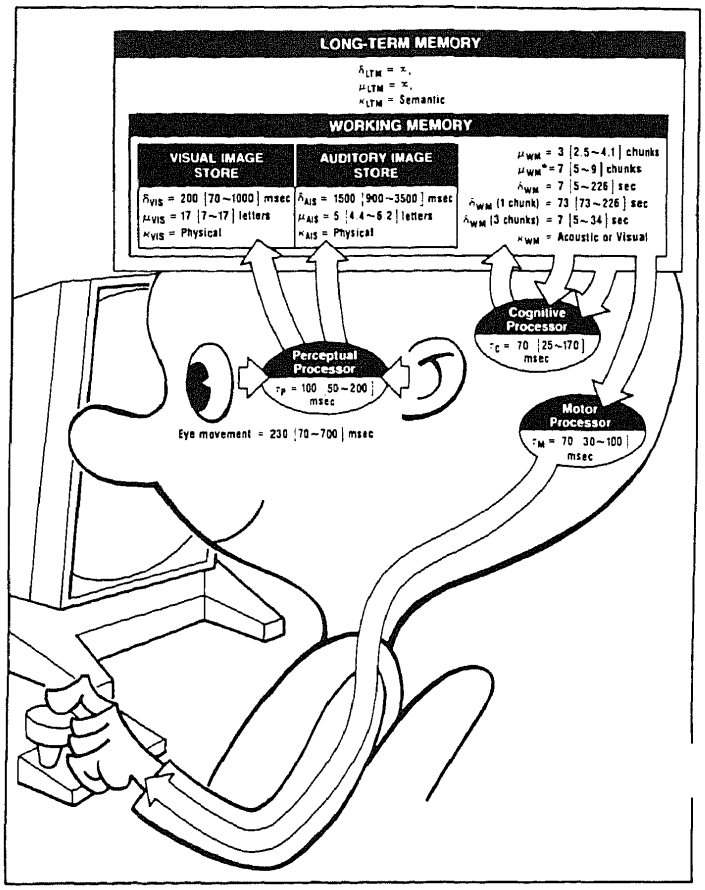
Les deux derniers chapitres présentent respectivement « dix hypothèses » pour les expériences en matière de facteurs humains, et des conseils pour la mise en place d'expériences de ce type.

Le livre de Ledgard *et al.* se présente comme une monographie, dans une série vouée en principe à la publication rapide de résultats sous une forme non définitive. Même ainsi présenté, cependant, je ne crois pas qu'il apporte beaucoup au lecteur.

3. Card, Moran, Newell

The Psychology of Human-Computer Interaction est très différent des deux ouvrages précédents. Card, Moran et Newell proposent une sorte de traité des applications de la recherche psychologique à l'interaction homme-machine; le volume (466 pages) est truffé de chiffres, de courbes, de références, mais se lit pourtant agréablement. Je ne cacherai pas que la lecture de cet ouvrage m'a passionné, ébranlé, même si je ne sais pas encore très bien quoi faire de toute la richesse des matériaux qu'il contient.

Card *et al.* sont, pourrait-on dire, des « modélisateurs de l'activité cognitive ». Leur but est de construire des descriptions adéquates de ce qui se passe dans le cerveau humain lorsqu'il participe à certaines activités, et tout particulièrement, bien sûr, à la communication homme-machine. Les modèles ainsi construits, et dûment validés par l'expérience, doivent permettre en retour de guider les concepteurs de nouveaux systèmes.



Le modèle de base est reproduit ici (c'est la figure 2.1 de l'ouvrage). Humanistes, frémissez ! Descartes, retourne-toi ! Ceux d'entre nous qui ont toujours été gênés par le C du sigle de la bienveillante Association qui édite cette revue en seront ici pour leurs frais : car c'est bien de cette Cybernétique chère à nos grands-parents qu'il s'agit, même si le terme n'apparaît jamais dans l'ouvrage. Donc, face à un ordinateur, chacun d'entre nous dispose d'un « processeur de perceptions » avec ses mémoires, auditive et visuelle; d'un « processeur de mouvements »; et d'un « processeur de connaissances », partageant avec le précédent une mémoire principale (« à court terme ») et une mémoire secondaire (« à long terme »). Les trois peuvent bien entendu fonctionner en pipeline; leurs caractéristiques physiques, données sur la figure, montrent que nous n'arrivons pas à la cheville d'un Vax ni même d'un Apple (τ est le temps de cycle d'un processeur, δ le temps d'accès à un élément en mémoire, μ la capacité, κ le type de codage).

Pour simple qu'il soit, ce modèle permet déjà de répondre à un certain nombre d'exercices que nous soumettons au lecteur (réponses après la bibliographie). Dans un jeu vidéo, une balle heurte une autre balle; de combien de temps dispose-t-on après la collision pour calculer le déplacement initial de la seconde si l'on veut

qu'il paraisse avoir été causé par le choc ? A quelle vitesse peut-on lire, combien de temps gagne-t-on par opération si l'on rapproche des touches numériques la touche « changement de fonction » d'une calculette ? etc.

Muni de ce modèle — qui est bien entendu discuté et justifié en détail —, les auteurs étudient ce qui se passe au cours d'une session avec un système interactif. Leur analyse se place à des niveaux de détail variables, au nombre de quatre (tâche à accomplir, fonction de base, argument d'une commande, touche du clavier) ; cette décomposition en plusieurs niveaux permet une étude plus fine que dans l'article publié par les mêmes auteurs il y a quelques années [Card 80]. Un des apports intéressants de l'ouvrage est un formalisme de type algorithmique (dont on peut regretter qu'il ne se conforme pas plus syntaxiquement, aux conventions des langages de programmation usuels) permettant de décrire les stratégies que les utilisateurs d'un programme interactif, un éditeur de textes par exemple, mettent au point de façon plus ou moins consciente. Cette description emploie elle aussi plusieurs niveaux : Buts, Opérateurs, Méthodes, Règles de Sélection (d'où le sigle GOMS, « Goals, Operators, Methods, Selection rules »). A titre d'exemple choisi au plus bas niveau (règles de sélection), la règle suivante décrit la stratégie observée auprès d'un certain nombre d'utilisateurs d'un éditeur de textes pleine page, utilisant une souris, pour désigner un objet auquel on souhaite appliquer une certaine opération (destruction, modification, etc.) :

Selection rules for GOAL : POINT-TO-TARGET

CHAR-POINT-RULE =
if VisualSearchTarget isa # CHARACTER
then CHOOSE (CHAR-POINT-METHOD)

WORD-POINT-RULE =
if VisualSearchTarget isa # WORD
then CHOOSE (WORD-POINT-METHOD)

TEXT-SEG-RULE =
if VisualSearchTarget isa # TEXT-SEG
then CHOOSE (TEXT-SEG-POINT-METHOD)

Les études extrêmement détaillées que contient l'ouvrage se rapportent à des sujets aussi divers que la comparaison de divers types d'éditeurs de textes, la conception de circuits intégrés, celle d'un système de photocomposition de presse, l'analyse des différents dispositifs de sélection sur écran, etc.

Tout ceci est remarquablement documenté. C'est tout un volet de la recherche en « psychologie cognitive » qui est ici synthétisé pour le plus grand profit des informaticiens, qui en général ne connaissent pas du tout ce domaine (à l'exception de quelques spécialistes d'intelligence artificielle, qui se sont intéressés aux modèles de la connaissance pour s'en inspirer dans leurs programmes). Or c'est aux informaticiens que, dans l'esprit des auteurs, l'ouvrage sera le plus utile, même s'il s'adresse aussi aux psychologues. Ils soutiennent en effet que la prise en compte des principes psychologiques est du ressort des premiers, non des seconds. Leur argument est qu'il n'est guère réaliste d'imaginer une situation où des psychologues et des ergonomes pourraient exercer un pouvoir réel sur la façon dont sont conçus et réalisés les systèmes informatiques : en effet, dans la pratique, les critères psychologiques et ergonomiques entrent inévitablement en conflit avec les autres critères qui président à la conception d'un produit logiciel (coût, efficacité, sécurité, etc.) et c'est l'ingénieur informaticien qui doit effectuer les compromis nécessaires, et donc disposer de la compétence correspondante. Card, Moran et Newell plaident donc pour l'inclusion de cours d'ergonomie dans les programmes d'enseignement du génie logiciel plutôt que pour la formation de psychologues ou ergonomes à spécialité informatique.

Pour impressionné que j'aie été par cette somme, j'ai quand même deux réserves sérieuses. J'ai tout d'abord été gêné par l'accent presque exclusif mis sur deux facteurs, le temps et la quantité d'information. Il y a chez les auteurs un côté un peu tayloriste de contrôleur posté à côté d'une ouvrière, son chronomètre à la main. Qu'un bon système doive permettre de faire vite les choses simples, certes ; mais qu'en est-il des autres qualités d'un logiciel interactif : facilité d'emploi et d'apprentissage, caractère « amical », etc. ? Est-il certain qu'ils soient tous réductibles à des mesures de temps et de volume ? Ne peut-on trouver des critères plus fins ? En écrivant cela, je sais que je me contredis en partie, puisque j'ai critiqué plus haut Shneiderman et les études qu'il cite pour leur manque de rigueur et de précision. Mais n'est-il pas possible de trouver des mesures qui, sans tomber dans le flou et le subjectif, décriraient autre chose que le temps nécessaire à chaque opération et le nombre d'éléments manipulés ?

Mon autre réserve est plus liée à la forme de l'ouvrage, qui est un travail de recherche. Quand on est soi-même un concepteur de systèmes interactifs, on ne peut manquer, me semble-t-il, d'être intéressé par ce livre et de sentir qu'on a beaucoup à y apprendre ; mais il reste à accomplir un effort important pour transposer tout ce matériau à la pratique et faire en sorte qu'il puisse vraiment influencer le travail quotidien de conception et de réalisation. Il manque, à côté du traité de Card, Moran et Newell — dont je pense qu'il restera comme un ouvrage de référence fondamental — un manuel plus appliqué, fournissant les éléments utiles à la pratique de l'informaticien.

Cela étant, je ne saurais recommander avec assez d'enthousiasme à tous ceux qui, s'intéressent aux systèmes interactifs et doivent en réaliser, qu'il s'agisse d'éditeurs de textes, de systèmes d'exploitation, d'outils de génie logiciel, de programmes de CAO ou d'EAO, etc., de lire sans tarder cet ouvrage.

[Gannon 77] John D. GANNON : *An experimental Evaluation of Data Type Conventions ;* Communications of the ACM, **20** (8), August 1977, 584-595.

[Hansen 71] W. J. HANSEN : *User Engineering Principles for Interactive Systems ;* Proceedings of the Fall Joint Computer Conference 39, 523-532.

[Shneidermann 83] Ben SHNEIDERMANN : *Direct Manipulation : A Step Beyond Programming Languages ;* Computer (IEEE), **16** (8), August 1983, 57-69.

[Weinberg 71] Gerald M. WEINBERG : *The Psychology of Computer Programming ;* Van Nostrand-Reinhold (New York), 1971.

RÉPONSES aux questions du paragraphe 3

100 millisecondes
652 mots/minute
90 millisecondes.