Guide ergonomique de conception des interfaces homme-machine

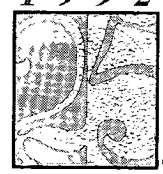
Dominique Scapin

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Rapports Techniques

1 9 9 2



ème anniversaire

N° 77

GUIDE ERGONOMIQUE DE CONCEPTION DES INTERFACES HOMME - MACHINE

Dominique L. SCAPIN

Octobre 1986

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

GUIDE ERGONOMIQUE DE CONCEPTION DES INTERFACES HOMME-ORDINATEUR

ERGONOMIC GUIDELINES FOR THE DESIGN OF HUMAN-COMPUTER INTERFACES

Dominique L. Scapin

Octobre 1986

AVERTISSEMENT

Ce guide comporte quatre parties: les recommandations, une bibliographie générale, un index permettant d'accéder au texte par mots-clefs et un aide-mémoire récapitulant certaines des questions ergonomiques qu'il faut se poser lors de la conception de l'interface.

La présente version de ce guide sera actualisée à mesure que de nouveaux résultats seront publiés dans la littérature. Cette version pourra aussi être améliorée dans le futur, en fonction des réactions qu'elle suscite, des questions qui seront posées, des suggestions, notamment de la part des concepteurs. L'étape suivante de ce travail sera de fournir une méthode détaillée de conception des interfaces.

Dans ce guide, notre objectif a été de rassembler une sélection de résultats ergonomiques aussi simplement et aussi clairement que possible, avec un certain souci d'exhaustivité, tout en restant très pratique, c'est-à-dire en présentant des recommandations directement applicables. Qui dit recommandations directement applicables dit aussi parfois recommandations qui s'expriment sans référence explicite aux tâches et applications particulières, ce qui dans certains cas conduit à des recommandations assez générales. En conséquence, il arrivera que certaines questions ergonomiques qui pourront se poser au concepteur, dans un environnement particulier, ne trouveront pas ici de réponses entièrement satisfaisantes. Trois raisons essentielles expliquent que tout guide de ce genre a des limites:

- certains systèmes sont d'un certain point de vue uniques, ce qui empêche d'appliquer strictement les connaissances obtenues sur d'autres systèmes;

- certaines questions de conception sont plus complexes que d'autres et/ou spécifiques d'une situation particulière, ce qui nécessite le recours à des spécialistes, et/ou l'utilisation de méthodes expérimentales;

- nombre de décisions de conception sont le fruit de plusieurs compromis entre plusieurs critères, comme entre temps d'apprentissage, fonctionnalité, temps de réponse, temps de transaction, clarté, brièveté, erreurs, etc. Quelquefois, améliorer la performance sur une dimension n'améliore pas forcément la performance sur une autre dimension (e.g. conflit entre facilité d'apprentissage et temps des transactions).

Ce guide pratique ne doit donc pas faire oublier que pour concevoir des logiciels ergonomiques, l'expertise de l'ergonome est souvent nécessaire, de même qu'il ne suffit pas de disposer d'une encyclopédie médicale pour soigner un malade.

I - INTRODUCTION

L' Interface Homme-Ordinateur (IHO) englobe tous les aspects des systèmes informatiques qui influencent la participation de l'utilisateur à des tâches informatisées. Les aspects immédiatement observables sont bien sûr l'environnement physique du travail et le matériel constituant le poste de travail. Ces aspects sont peut être les mieux connus, car étudiés depuis plus longtemps, mais ce ne sont pas les seuls, ni les plus importants. D'autres aspects, plus subtils, concernent les diverses façons dont les informations sont stockées et manipulées. Ceci inclut les procédures informatiques, mais aussi les outils annexes, tels les documents papier, les aides au travail, etc.

Par opposition à d'autres machines qui représentent des extensions du corps humain, par exemple qui fournissent des capacités et une puissance supplémentaires, l'ordinateur, et en particulier le logiciel, représente plutôt une extension du cerveau humain, en faisant appel à des processus plus cognitifs (e.g., perception, mémoire, language, résolution de problèmes, prise de décision, etc.).

En conséquence, pour améliorer l'IHO, il convient de s'intéresser plus particulièrement aux processus cognitifs humains. Utilisant les résultats et les méthodes de l'ergonomie, l'objectif de ce guide est de fournir un certain nombre de recommandations pour la conception de logiciels mieux adaptés aux processus cognitifs des utilisateurs. L'objectif est de fournir des éléments

permettant d'améliorer la communication utilisateur-ordinateur au même titre que l'on s'assure de la performance d'autres aspects des systèmes, tels que la qualité des algorithmes, le temps de réponse, la sécurité et la fiabilité.

Ce qui est spécifique à l'ergonomie des logiciels par rapport à d'autres aspects plus classiques de l'ergonomie est que la conception du logiciel établit le contenu des informations disponibles à l'utilisateur ainsi que les relations visuelles entre ces informations. En combinaison avec, ou à la place du matériel, le logiciel établit aussi la séquence des actions que l'utilisateur doit effectuer ainsi que le feed-back concernant ces actions. Par opposition à un environnement papier ou à des dispositifs plus classiques de présentation d'informations et de commandes ("displays and controls", "knobs and dials") la tâche de l'utilisateur est intimement dépendante du logiciel. En conséquence, l'impact de l'ergonomie dans le développement du logiciel est très important, puisque cela concerne l'ensemble de l'activité de l'utilisateur. L'ergonomie ne s'intéresse pas seulement aux aspects de surface de l'IHO, mais à tout ce qui peut influencer l'activité de l'utilisateur. Il ne s'agit en effet pas seulement de définir la taille des caractères, mais aussi de définir quelles informations doivent être fournies.

Pour ces raisons, l'ergonomie du logiciel a eu historiquement (et a toujours) une part importante dans la conception et l'évaluation de systèmes critiques tels que le contrôle du trafic aérien, les applications militaires et les centrales nucléaires. Dans d'autres environnements qui ne posent pas de problèmes aussi aigus de sécurité, l'ergonomie des logiciels n'est devenue une discipline essentielle que depuis dix ans. De nombreux organismes, entreprises, qui produisent ou simplement utilisent des systèmes informatiques incluent maintenant l'ergonomie dans leurs travaux de conception et d'évaluation de systèmes, mais aussi dans leur stratégie publicitaire et de marketing. Dans ces environnements, les améliorations attendues sont bien sûr liées à la satisfaction des utilisateurs et à une meilleure acceptabilité, mais aussi à une meilleure efficacité des opérations, à des économies de temps et d'argent, et finalement à des profits plus élevés pour les entreprises.

L'objectif de ce guide est de recenser les divers problèmes d'ordre ergonomique posés par les Interfaces Homme-Ordinateur et d'y apporter un

certain nombre de propositions de solutions. Ces recommandations sont issues en grande partie des résultats des recherches en ergonomie du logiciel mais aussi d'une pratique de conseil sur le terrain. Ce guide est donc entaché d'une certaine subjectivité, à la fois quant au choix des recommandations (certaines ne sont pas mentionnées parce que trop spécifiques ou parce qu'elles manquent d'étayage expérimental ou bien encore parce qu'elles sortent du cadre fixé) mais aussi quant au contenu de certaines règles. En effet, certains choix ont été faits quand les résultats de plusieurs études étaient en désaccord.

Cependant, ces précautions prises, il convient de signaler que ce guide correspond à des demandes souvent réitérées de la part des concepteurs qui souhaitent disposer de quelques règles de conception.

En conséquence, malgré la répugnance de certains ergonomes à se montrer normatifs, nous avons pris le parti d'énoncer un certain nombre de recommandations. Même si ces recommandations ne sont pas complètes, même si elles peuvent être remises en cause par de futurs travaux scientifiques, même si on ne peut s'en contenter toujours et s'il faut bien souvent faire appel à l'expert en ergonomie, énoncer ces recommandations a le mérite de permettre l'introduction de considérations ergonomiques dans des entreprises qui de toute façon ne pourraient faire appel aux ergonomes chaque fois qu'elles concoivent un produit logiciel, et qui préfèrent avoir des recommandations qui "marchent" dans 75% des cas plutôt que pas de recommandation du tout.

Rappelons aussi que ce guide ne concerne que certains aspects de l'interface (i.e., les aspects indépendants de tâches spécifiques), qu'il y a certains compromis à faire entre les diverses recommandations et que l'aspect combinatoire des règles restera cependant assez complexe à manipuler (en effet, à un point donné de la conception d'une transaction, il convient d'examiner de nombreuses règles, souvent inter-dépendantes). En fait, une contribution ultérieure à ce guide pourra être de formaliser de façon exhaustive la combinatoire des règles afin qu'elles soient appliquées de façon concomitante, notamment à des fins évaluatives (la technique d'implémentation pouvant, par exemple, prendre la forme d'un système expert).

II - DEFAUTS DE CONCEPTION DE L'IHO

1 - Caractéristiques

Un certain nombre d'exemples de déficiences caractérisent la conception des logiciels. En particulier, les concepteurs ont tendance à:

- manquer de connaissances préalables des tâches et des utilisateurs (e.g., transactions ne correspondant pas à l'expérience des utilisateurs, séquences de commandes non conformes aux séquences des actions pour la tâche, informations ou fonctions manquantes ou superflues, etc.);

- manquer de méthodologie de conception de l'IHO (ou plutôt utilisation de méthodes informatiques fonctionnelles n'incluant pas la prise en compte de l'opérateur humain);

- concevoir selon une orientation fonctionnelle plutôt qu'opérationnelle (i.e., en fonction d'une logique de fonctionnement plutôt que d'une logique d'opération, en fonction de la structure des fichiers plutôt que de la structure des informations pour l'utilisateur);

- ne pas évaluer précisément les conséquences combinatoires des transactions de dialogue (e.g., conception indépendante des transactions, conception en termes d'écrans plutôt que de transactions plus globales, pas d'évaluations sur des scénarios réalistes de tâches, etc.);

- manquer d'homogénéité dans la conception (aussi bien en ce qui concerne les commandes ou les présentations d'informations, on peut relever des incohérences dans la dénomination, la structure, l'ordre des items);

- ne pas prévoir les erreurs humaines (e.g., fonctions destructrices non protégées, erreurs de syntaxe non prévues, alternatives de choix non prises en compte, etc.);

- concevoir selon des critères de performance des systèmes plutôt que des critères liés aux objectifs des utilisateurs et aux contraintes de la tâche;

- fournir toutes les fonctions imaginables plutôt qu'un ensemble plus réduit de fonctions essentielles;

- fournir toutes les informations disponibles (au cas où elles se perdraient!) plutôt que seulement celles nécessaires à la tâche;

- enfin, considérer l'ordinateur comme une fin en soit plutôt qu'un moyen pour une fin. L'ordinateur est en effet un outil permettant d'accomplir une tâche, au même titre que d'autres outils également disponibles (e.g., crayon, formulaires, téléphone, etc.). La valeur de ces outils doit être jugée en fonction de la façon dont ils aident les utilisateurs à atteindre leurs objectifs, et non pas en fonction de leur nouveauté, de leur vitesse ou de leur sophistication.

2 - Causes et conséquences

Les défauts de conception de l'IHO résultent en général de diverses erreurs qui tiennent des croyances. Deux d'entre elles sont encore trop communes.

La première erreur est de croire que des améliorations pour l'utilisateur seront issues uniquement des progrès technologiques, alors que chaque nouvelle technologie (e.g., les écrans ou la commande vocale) fait surgir de nouveaux problèmes (e.g. disposition spatiale des informations ou entraînement du locuteur) par rapport à la technologie précédente (respectivement les télétypes ou les claviers).

La seconde erreur est de penser que pour concevoir des logiciels ergonomiques, il suffit d'y réfléchir un peu. Or on sait bien que l'introspection dans ce domaine, surtout à propos de phénomènes cognitifs complexes, ne donne pas de bons résultats.

Deux des conséquences de ces croyances sont que souvent les concepteurs ne connaissent pas suffisamment bien les tâches qu'ils sont

supposés informatiser, et qu'ils ne réalisent pas l'effort, ou ne connaissent les méthodes appropriées nécessaires pour obtenir cette connaissance. Une autre conséquence est que les concepteurs ont souvent tendance à fournir aux utilisateurs tout ce à quoi ils peuvent penser. Malheureusement, fournir dix mauvais outils n'est pas meilleur que de n'en proposer qu'un seul qui soit bien adapté.

L'organisation des équipes de conception ainsi que les contraintes de temps sont aussi des facteurs importants dans les erreurs de conception. Le fait que le logiciel soit implémenté par des individus qui ne partagent pas nécessairement la même vue des procédures opérationnelles désirées peut conduire à des interfaces non homogènes, voire incompatibles. Quand un utilisateur doit effectuer des tâches avec des procédures différentes, ou pire, quand il doit effectuer différemment des transactions conceptuellement similaires au sein d'une même tâche, une contrainte non nécessaire lui est imposée pour l'apprentissage et l'utilisation du système. Quand plusieurs systèmes sont développés au sein d'une grande entreprise et quand la conception de l'IHO est établie indépendamment pour chaque système, le résultat est souvent d'imposer des comportements incompatibles à des utilisateurs qui pourront avoir à utiliser plusieurs systèmes, ou tout simplement communiquer avec d'autres utilisateurs qui ont besoin de coopérer vers le même objectif.

Les défauts de conception de l'IHO conduisent à une performance dégradée des systèmes. Les utilisateurs peuvent parfois compenser une mauvaise conception par des efforts supplémentaires. Cependant, il y a une limite à l'adaptation des utilisateurs à une mauvaise interface. L'effet négatif cumulé résultant de plusieurs erreurs de conception peut conduire à des dysfonctionnements des systèmes, à des performances insatisfaisantes, et à des plaintes des utilisateurs.

Cela peut aussi conduire à:

- une non-utilisation du système et au recours à d'autres sources d'information;

- une diminution de l'utilisation quand celle-ci est optionnelle et à une régression à des procédures manuelles;

- mauvaise utilisation, contournement des règles du système pour court-circuiter les difficultés rencontrées pour réaliser une opération;

- utilisation partielle, utilisation seulement d'un sous-ensemble des capacités du système;

- emploi d'un intermédiaire entre l'utilisateur et le système (conduite typique des managers);

- modification de la tâche;

- activités compensatoires, opérations supplémentaires ou détournées;

- frustation, désintérêt, rejet, taux d'erreurs élevés, performance faible, etc.

Cela signifie bien entendu des opérations interrompues, du temps, des efforts et des investissements perdus, et l'échec vis-à-vis des avantages potentiels de l'informatisation. Par ailleurs, le passage de procédures manuelles à l'informatique peut aussi signifier en fait deux fois plus de temps nécessaire pour mener des tâches à leur terme en raison de déficiences dans la conception de l'IHO (en effet, dans certains cas, la tâche est doublée, i.e., qu'il y a conservation de la procédure manuelle en plus de la procédure informatisée).

3 - Recours à l'ergonomie

La conception actuelle de la plupart des IHO peut être considérée comme un art plutôt qu'une science, reposant sur des opinions ou jugements individuels, plutôt que sur l'application systématique de connaissances. C'est cette connaissance, à la fois en termes de méthodes et de résultats que l'ergonomie peut apporter à la conception de logiciels. En effet, l'ergonomie des logiciels n'est pas seulement du sens commun, c'est une science appliquée.

Dans l'absence de guidage effectif, la conception et l'implémentation de l'IHO peut devenir l'apanage de programmeurs non familiers avec les exigences opérationnelles de la tâche. L'IHO est alors produite lentement, la détection et la correction des erreurs de conception (quand cela est fait) intervient seulement après l'implémentation initiale des systèmes, quand les modifications du logiciel sont difficiles et coûteuses. La mise en oeuvre de l'ergonomie devrait donc se faire à chaque étape de la conception, le plus tôt possible, plutôt qu'a posteriori, lors d'une contribution évaluative extérieure.

III - LES PRE-REQUIS DE LA CONCEPTION

Avant de concevoir des logiciels d'interface, deux éléments des systèmes Homme-Machine doivent être bien connus: les utilisateurs potentiels et la tâche.

1 - Les utilisateurs

Tout d'abord le concepteur doit être conscient qu'un nouveau système ne peut être adapté à la fois à tous les utilisateurs potentiels. Pour qu'il soit adapté à une population particulière, les caractéristiques de ces utilisateurs doivent être bien connues. En effet, des différences de population en termes de capacités, expérience, formation, etc., peuvent influencer la performance des utilisateurs sur des tâches informatisées. Les distinctions habituelles concernent par exemple les managers qui ont besoin de systèmes pertinents, faciles à utiliser; les spécialistes qui ont besoin de systèmes sur mesure, avec un recours possible à des programmeurs, les administratifs qui ont besoin de systèmes qui limitent les tâches ennuyeuses et répétitives.

Une des difficultés majeures est qu'un système aura des utilisateurs avec diverses caractéristiques qui vont évoluer à mesure que ces derniers acquièrent de l'expérience avec le système. Idéalement, un objectif de conception devrait donc être de construire des interfaces dont les éléments importants changent en fonction de l'acquisition d'expérience des utilisateurs, i.e., des systèmes possédant divers niveaux d'interface correspondant aux divers niveaux d'expérience.

Pour déterminer les caractéristiques de l'IHO, les concepteurs devraient connaître le détail des besoins des utilisateurs potentiels. En fait, les utilisateurs eux-mêmes peuvent aider à la définition des caractéristiques de l'IHO, dans la mesure où les bonnes méthodes sont utilisées, même si les utilisateurs ne sont pas familiers avec l'état de l'art des technologies informatiques et même s'ils ne peuvent pas toujours penser à la meilleure manière de tenir compte de leurs besoins. Par le moyen d'interviews et de mises en situation d'essai, on peut s'assurer avec les utilisateurs les plus expérimentés que toutes leurs exigences sont prises en compte. Par ailleurs, on sait que les utilisateurs potentiels seront d'autant plus enclins à accepter un système qu'ils ont été associés à l'établissement du cahier des charges et s'ils observent qu'un certain nombre de leurs suggestions ont été implémentées.

En résumé, il est souhaitable d'obtenir le concours des utilisateurs, mais il faut savoir que ces derniers ont des limitations.

Pour obtenir des informations exactes et appropriées auprès des utilisateurs, il est essentiel de s'assurer du concours de l'ergonome. L'ergonome peut en effet être considéré comme un médiateur entre le concepteur et l'utilisateur: il connaît les méthodes permettant de réunir les informations et exigences pertinentes pour la tâche de l'utilisateur et a l'expérience des différentes manières dont ces exigences peuvent être traduites dans la conception du logiciel.

Les utilisateurs novices ou naïfs sont probablement la population la plus large des utilisateurs de l'informatique. Ces derniers devraient être amenés à considérer l'ordinateur comme une machine qui obéit à des instructions exprimées selon un ensemble de règles très strictes, et non pas comme quelque chose de magique (souvent les utilisateurs attribuent à l'ordinateur plus d'intelligence qu'il n'en a en réalité). Il n'est pas nécessaire pour la plupart des utilisateurs de connaître en détail la nature ou le fonctionnement de l'ordinateur, de même que pour conduire une automobile, il n'est pas nécessaire de connaître dans le détail le fonctionnement du moteur à explosion.

Un point de vocabulaire: dans ce guide, on utilisera le terme expérimenté en se référant à l'expérience en informatique, particulièrement à l'expérience de

l'interface et non pas (sauf cas précisés) en se référant à l'expérience de la tâche, notamment non informatisée.

Avant de présenter plus loin des recommandations plus précises, on peut déjà identifier un certain nombre de règles de conception du logiciel pour les utilisateurs naifs:

- l'initiative doit venir du logiciel;

- chaque entrée doit être brève;

- les procédures d'entrée doivent être conformes aux attentes des utilisateurs;

- elles ne doivent pas requérir de formation particulière;

- le logiciel pour certaines entrées, notamment à risque (i.e., ayant des conséquences importantes et/ou irréversibles), doit requérir une confirmation de la part de l'utilisateur;

- les messages doivent être clairs et sans équivoque, et ne pas contenir d'information superflue;

- le contenu des messages doit être tel qu'il offre un nombre limite d'options à l'utilisateur, limitant ainsi le nombre de décisions à prendre;

- l'utilisateur doit être capable de contrôler le rythme du dialogue;

- l'utilisateur doit avoir la possibilité d'obtenir facilement une aide humaine (e.g., de la part d'un utilisateur expérimenté ou d'un technicien).

De façon générale, le concepteur doit garder en tête la façon dont le système va apparaître à l'utilisateur. Il doit connaître le séquencement logique des activités de l'utilisateur, basé sur l'analyse du travail. Quelle que soit la complexité apparente d'une transaction, il doit être possible de la découper en une série d'étapes plus simples.

La prise en compte des besoins des utilisateurs est très importante. Des lacunes dans ce domaine, qui produiront des systèmes inadaptés, conduisent à divers comportements des utilisateurs, tels que ceux qui ont été mentionnés précédemment.

2 - La tâche

Connaître précisément la tâche est essentiel pour la conception du logiciel. De nombreuses méthodes sont disponibles pour étudier et décrire les tâches avec ou sans sytèmes informatiques, avec l'objectif d'établir les exigences liées aux tâches ou pour évaluer des systèmes existants ou des prototypes.

On ne décrira pas ici ces méthodes. De telles descriptions sont en effet disponibles dans les manuels d'ergonomie, section analyse du travail. Cependant, il est important de rappeler à quoi servent ces techniques.

De façon générale, les techniques d'analyse du travail sont des moyens d'obtenir les données nécessaires pour alimenter les décisions mises en oeuvre dans la conception du logiciel. L'objectif est de définir, du point de vue de l'utilisateur, les exigences auxquelles le logiciel doit se conformer.

Cela concerne aussi bien le recensement des informations et fonctions constituant les tâches que le séquencement opérationnel entre événements, données, décisions, actions, etc. De telles analyses permettent la constitution de diagrammes de fluence, d'organigrammes et de scénarios qui seront des outils pour les décisions de conception, les simulations, et les évaluations. Les données recueillies concernent des aspects tels que les entrées/sorties, l'allocation des fonctions entre l'homme et l'ordinateur, la séquence des opérations, les caractéristiques et l'organisation des données, le niveau d'apprentissage requis, etc.

Ces analyses devraient faire partie d'un processus itératif aux différentes étapes du développement et de l'implémentation du logiciel. Ce n'est qu'en étant partie intégrante du processus de conception que ces analyses permettront d'éviter les erreurs de conception liées à une méconnaissance de la tâche et des

utilisateurs, ou à un manque de méthodologie adaptée. De simples analyses sur le terrain ainsi que l'utilisation de techniques appropriées de description de la tâche et du dialogue permettent souvent d'éviter des erreurs de conception grossières.

Il faut souligner qu'une certaine prudence est nécessaire dans la pratique des techniques d'analyse du travail. En effet, celles-ci (e.g., observations, interviews, questionnaires, incidents acritiques, etc.) requièrent une formation et une certaine expérience pour être pertinentes et efficaces. Il y a en effet par exemple la tentation chez l'analyste novice de décrire certains événements ou certaines caractéristiques de la tâche qui sont sans importance ou traité(e)s hors de proportion, alors que d'autres aspects plus importants pour la conception de l'interface sont laissés de côté. Il y a aussi parfois la tentation de satisfaire le client plutôt que de mettre l'accent sur des problèmes critiques, ou bien encore de quantifier des données, des temps, sans la méthodologie statistique appropriée. Là encore, le recours au spécialiste en ergonomie est souhaitable.

IV - CRITERES DE CONCEPTION ERGONOMIQUE DE L'INTERFACE

De façon générale, on peut dire qu'il est essentiel de respecter les objectifs, connaissances, représentations et méthodes des utilisateurs. Les analyses mentionnées précédemment devraient avoir clairement identifié ces paramètres, en particulier les objectifs de la tâche, la façon dont les informations sont organisées et traitées, la façon dont les décisions sont prises, etc.

Comme on l'a dit précédemment, nombre de décisions de conception dépendent intimement des caractéristiques spécifiques de la tâche à informatiser. Cependant, pour des aspects plus généraux de la conception des interfaces, on peut identifier un certain nombre de principes ergonomiques à respecter.

Ces principes concernent: la compatibilité, l'homogénéité, la concision, la flexibilité, le feed-back (et le guidage), la charge informationnelle de l'utilisateur, le contrôle explicite et la gestion des erreurs.

1 - Compatibilité

Le principe de compatibilité repose sur le fait que les transferts d'information seront d'autant plus rapides et plus efficaces que le recodage d'information est réduit. La mise en application de ce principe conduit aussi bien à rendre des écrans compatibles avec des supports papier, que des dénominations de commandes compatibles avec le vocabulaire de l'utilisateur, etc.

2 - Homogénéité

Le principe d'homogénéité repose sur le fait que la prise de décision, le choix de solutions, le rappel, etc., peuvent se répéter de façon d'autant plus satisfaisante que l'environnement est constant. La mise en oeuvre de ce principe permet notamment le développement de procédures quasi-automatiques et le transfert à des procédures nouvelles de principes acquis lors de la mise en oeuvre avec succès de procédures connues. Cela signifie par exemple qu'il faut s'assurer qu'on utilise des séquences de commandes identiques pour arriver au même résultat (notamment d'une transaction à une autre) et que les labels, prompts et autres catégories d'informations sont localisés au même endroit d'un écran à l'autre.

3 - Concision

Le principe de concision repose sur l'existence de limites en mémoire à court terme de l'opérateur humain. Il convient donc de réduire la charge mnésique des utilisateurs. L'application de ce principe consiste par exemple à éviter à l'utilisateur de mémoriser des informations longues et nombreuses, ou des procédures trop longues. L'ordinateur doit être utilisé autant que possible comme mémoire externe, au même titre que les aides au travail.

4 - Flexibilité

La flexibilité est une exigence liée à l'existence de variations au sein de la population des utilisateurs. Il est en effet souhaitable que le logiciel

comporte différents niveaux et qu'il prenne en considération l'acquisition d'expérience des utilisateurs.

5 - Feed-back et guidage

La nécessité de feed-back repose sur l'influence de la connaisance du résultat sur la qualité de la performance. Le feed-back doit être aussi immédiat que possible (notamment pour le feed-back d'entrée d'informations ou de commandes). Les résultats des actions de l'utilisateur doivent toujours être répercutés de façon explicite. De manière générale, une attention particulière doit être apportée au feed-back et au guidage. Un niveau acceptable de performance ne pourra être atteint que si l'utilisateur est informé rapidement et de façon adéquate sur le succès de ses actions. L'utilisateur doit toujours savoir où il se trouve dans une séance de dialogue, ce qui a été fait, et avoir toujours un moyen de poursuivre le dialogue.

6 - Charge informationnelle

La prise en compte de la charge informationnelle de l'utilisateur est essentielle dans la mesure où la probabilité d'erreur humaine augmente dans les situations à charge élevée. Par exemple, il convient de minimiser le nombre d'opérations à effectuer par l'utilisateur ainsi que les temps de traitement. Un certain nombre de décisions permettent par exemple de réduire les temps d'entrée (touches fonctions, valeurs par défaut, etc.). Par ailleurs, l'utilisateur ne doit pas avoir à attendre trop longtemps que l'ordinateur effectue les ordres donnés.

7 - Contrôle explicite

Le principe de contrôle explicite signifie que même si c'est le logiciel qui a le contrôle, l'interface doit apparaître comme étant sous le contrôle de l'utilisateur et surtout, en règle générale, exécuter des opérations uniquement à la suite d'actions explicites de l'utilisateur.

8 - Gestion des erreurs

Une autre exigence est de fournir aux utilisateurs des moyens de corriger leurs propres erreurs. Un système bien conçu doit réduire les occasions propices à l'erreur, augmenter la capacité de l'utilisateur à détecter ses propres erreurs et fournir des moyens aisés de les corriger.

V - RECOMMANDATIONS

Dans ce qui va suivre, les recommandations seront présentées en fonction des trois éléments importants de l'IHO:

A - le dialogue, c'est-à-dire la communication dans les deux sens entre l'utilisateur et l'ordinateur.

B - les entrées, i.e., la communication de l'utilisateur vers l'ordinateur.

C - les sorties, i.e., la communication de l'ordinateur vers l'utilisateur.

Dans la section réservée au dialogue, après avoir discuté des propriétés générales des dialogues, on présentera les types de dialogue les plus répandus, ainsi que des considérations sur les procédures de dialogue.

La section sur les entrées concernera respectivement l'entrée d'informations, la dénomination, et les séquences de commandes.

Dans la section consacrée aux sorties, on évoquera le temps de réponse, les étapes de la conception des écrans, les problèmes de codage, pour terminer sur des recommandations plus spécifiques sur les affichages.

A - LE DIALOGUE

1 - Propriétés générales des dialogues

a - Initiative

L'initiative fait référence au fait que c'est l'utilisateur ou bien l'ordinateur qui dirige les transactions au sein du dialogue. Si l'ordinateur pose des

questions, présente des alternatives, et que l'utilisateur y répond, c'est un dialogue à l'initiative de l'ordinateur. Si au contraire l'utilisateur entre directement des commandes sans un tel guidage, le dialogue est alors à l'initiative de l'utilisateur. Bien entendu il existe divers degrés d'initiative.

On peut dire cependant de façon générale que des dialogues à l'initiative de l'ordinateur sont préférables pour des utilisateurs inexpérimentés ou occasionnels. De tels dialogues permettent en effet de mettre en oeuvre des processus de reconnaissance, plus aisés que des processus de rappel. Ils fournissent à l'utilisateur un modèle du système, et permettent donc l'utilisation du système par des opérateurs qui n'ont pas encore intériorisé ce modèle.

Par contre, un tel type de dialogue, qui ne permet pas d'anticiper, de sauter des étapes, ne peut être acceptable pour un utilisateur expérimenté que dans la mesure où le nombre de transactions est relativement réduit et que le temps de réponse est court. Un dialogue lent à l'initiative de l'ordinateur est en effet très perturbateur pour un utilisateur expérimenté.

En fait, pour la plupart des systèmes, il est souhaitable de permettre aux utilisateurs de choisir l'un de ces modes.

b - Flexibilité

La flexibilité peut se référer à deux notions différentes. La première est celle qui a été utilisée auparavant et qui correspond en fait à la capacité d'adaptation du logiciel à diverses sous-populations différenciables selon leur niveau d'expérience. Ce type de flexibilité, comme il a été dit, est souhaitable.

La seconde est plutôt une flexibilité "interne" et correspond au nombre de façons différentes (procédures, options, commandes, etc.) mises à la disposition de l'utilisateur pour atteindre un même objectif. Une grande flexibilité peut être obtenue par exemple en fournissant de nombreuses commandes, en permettant à l'utilisateur de définir ou redéfinir de nouvelles commandes. Des résultats montrent que les utilisateurs peu expérimentés tendent à utiliser des méthodes bien connues pour effectuer leurs opérations

plutôt que des méthodes parfois plus efficaces mais moins bien connues et que l'existence de dialogues très flexibles diminue la performance (notamment en augmentant le nombre d'erreurs) chez des utilisateurs naïfs. En conséquence, des dialogues très flexibles ne sont pas recommandés, excepté pour des utilisateurs expérimentés.

En somme, il est bon qu'un dialogue comporte plusieurs niveaux d'expérience, mais il est souhaitable que certains de ces derniers ne soient flexibles (au sens ci-dessus) que s'ils sont destinés à des utilisateurs expérimentés.

c - Complexité

La complexité est liée à la flexibilité: elle se caractérise en effet par le nombre d'options disponibles à l'utilisateur à un point donné de la transaction. Une complexité faible peut être obtenue en utilisant peu de commandes ou bien en subdivisant ces commandes de façon à ce que l'utilisateur n'ait à en sélectionner qu'une sous-partie à tout instant du dialogue. On peut penser qu'il existe un niveau optimal de complexité, pour une tâche et un utilisateur particuliers. Des études montrent qu'un nombre important de commandes redondantes ou non pertinentes handicape la performance, mais qu'une extrême simplification du dialogue par hiérarchisation à outrance est également mauvaise. C'est semble-t-il une propriété difficile à définir à l'avance, notamment en raison de ses fortes interactions avec les caractéristiques des utilisateurs et celles de la tâche. Ceci est un exemple des limites de ce guide: en effet, la complexité se règle en fonction d'éléments spécifiques d'une transaction particulière, au sein d'une tâche particulière.

d - Puissance

La puissance représente le volume de traitements effectués par l'ordinateur en réponse à une seule commande de l'utilisateur. Dans un dialogue avec des commandes puissantes, l'utilisateur peut accomplir avec une seule commande ce qui nécessiterait plusieurs commandes moins puissantes. Dans un certain nombre d'applications, des commandes

puissantes peuvent être souhaitables (e.g., opérateurs matriciels pour un logiciel de mathématiques). Le problème est qu'en général l'existence de commandes très puissantes (et donc par extension, souvent très spécifiques), réduit la généralité d'un système, son adaptabilité à d'autres tâches. Ceci est un facteur prépondérant dans le rejet des systèmes par les managers et le personnel technique (notamment du fait que leurs activités n'étant pas routinières, ils peuvent avoir besoin de faire appel à des fonctionnalités légèrement différentes). De plus, fournir des commandes puissantes en plus d'un ensemble de commandes de base, plus simples, tend à augmenter la complexité du dialogue. Une solution possible est de faire une partition du dialogue de telle sorte que par exemple les utilisateurs peu expérimentés disposent seulement d'un sous-ensemble des commandes, à un instant donné.

e - Charge informationnelle

La charge informationnelle est le degré selon lequel l'interaction sollicite les ressources de mémoire et de traitement de l'utilisateur. Dans la plupart des tâches, la performance des utilisateurs est influencée négativement quand la charge informationnelle est trop élevée ou trop faible. Bien entendu, une part importante de cette charge est fonction de la tâche elle-même et de la familiarité de l'opérateur vis-à-vis de sa tâche. Cependant la charge informationnelle est aussi influencée par les caractéristiques du dialogue. On peut mesurer cette charge de façon empirique ou bien l'estimer. Il est possible de la faire varier en intervenant sur les modalités d'affichage, les types de canaux sollicités, la puissance des commandes, l'utilisation de valeurs par défaut, le type et la structure des langages de commande et d'autres aspects de l'interface sur lesquels on reviendra.

2 - Quelques types de dialogue

Un dialogue peut être défini comme le type de relation qui existe entre les entrées et les sorties, i.e., entre les modes d'entrée et les modes d'affichage. Chacun de ces modes est très varié, particulièrement si l'on considère les moyens physiques d'entrée (clavier, light pen, commande vocale, écran tactile, souris, etc.). Ici, on ne mentionnera les modes d'entrée

que pour mémoire (voir la section sur les Entrées), pour évoquer les types de dialogue les plus communs, avec une attention particulière pour les touches fonctions, les menus, et les langages de commande.

Remarquons que la plupart des recommandations fournies plus loin sont indépendantes des modes physiques d'entrée ou de sortie. En particulier, les recommandations ultérieures sur les entrées et les sorties s'appliqueront en majeure partie aux divers types de dialogue présentés ci-après.

a - Question/réponse

Le logiciel pose une série de questions auxquelles l'utilisateur répond. Ce type de dialogue est complètement à l'initiative de l'ordinateur. Ce type de dialogue est probablement celui qui conduit le moins à des erreurs de la part des utilisateurs. Cependant, ce type de dialogue devient vite lourd et ennuyeux à mesure que l'utilisateur acquiert de l'expérience avec le système. Ce type de dialogue ne peut évidemment être utilisé que lorsque les données à entrer sont bien connues et que leur ordre peut être contraint (i.e., déterminé à l'avance).

Le mode question/réponse est un type de dialogue adapté quand les informations à obtenir ne peuvent être organisées sous forme de liste ou ne peuvent être facilement codées. Pour chaque réponse attendue de l'utilisateur, il est souhaitable de fournir un exemple de syntaxe correcte et de contenu approprié.

b - Remplissage de formes

Le logiciel présente des formes avec des espaces. C'est aussi un type de dialogue à l'initiative de l'ordinateur, mais qui est plus rapide que le type question/réponse car l'utilisateur peut entrer plusieurs réponses lors de la même transaction. Quand l'entrée d'informations concerne plutôt des paramètres que des commandes, ce type de dialogue est souvent le meilleur.

Ce type de dialogue est souhaitable notamment quand une certaine flexibilité est nécessaire pour l'entrée d'éléments optionnels. De plus, c'est

utile quand l'utilisateur doit entrer des commandes ou des informations déjà écrites ou imprimées sous forme papier. Dans le cas d'utilisation de formulaires papier, la disposition de l'écran doit être, autant que possible isomorphe à ces derniers (dans la mesure ou ceux-ci sont eux-mêmes bien adaptés).

Pour réduire le temps de réponse, quand le type de terminal le permet, il est souhaitable d'enregistrer les entrées de l'utilisateur dans un buffer et d'effectuer le transfert des informations en une seule fois (i.e., la page de formulaire entière) plutôt qu'item par item.

Quand l'utilisation d'un tel dialogue est fréquente, des fonctions de tabulation sont nécessaires.

Un autre avantage de ce type de dialogue est que, dans certaines circonstances, il est possible de détecter facilement certaines erreurs syntaxiques en établissant des contraintes sur les champs d'entrée.

c - Menus

L'ordinateur présente une liste de choix possibles parmi lesquels l'utilisateur effectue sa sélection. A nouveau à l'initiative de l'ordinateur, ce type de dialogue convient aussi bien à la construction de commandes qu'à l'entrée de paramètres. C'est d'un usage assez pratique si le temps de réponse est satisfaisant et si des moyens d'entrée par pointage direct sont utilisés. Une contrainte des menus est bien évidemment qu'il n'est pas possible d'entrer des données arbitraires (i.e., des donnée non définies précisément à l'avance, souvent en raison de leur nombre élevé et de leur variabilité, comme pour certains paramètres, ou certaines informations sur l'environnement).

L'utilisation de menus est conseillée quand le nombre des commandes est si élevé qu'on peut prévoir qu'il dépassera les capacités mémoire des utilisateurs. Dans ce cas les menus jouent le rôle d'aide mémoire. Cela permet d'ailleurs de passer ensuite à des langages de commande. En effet, après l'entrée fréquente de commandes, l'utilisateur acquiert une certaine

expérience d'utilisation qui lui permet ensuite de passer à un type de dialogue qui soit plus à sa propre initiative (dans la mesure où le logiciel dispose de plusieurs niveaux de dialogue).

De façon générale, les menus sont appropriés quand on souhaite s'assurer d'une bonne qualité initiale des entrées (i.e., limiter les erreurs lors de premières utilisations), quand les utilisateurs ne sont pas familiers avec toutes les fonctions du système.

Notamment pour des utilisateurs inexpérimentés, chaque menu ne devrait requérir qu'une seule sélection à la fois, laquelle devrait se faire par pointage direct.

Cependant, occasionnellement quand un choix parmi des options dont la longueur est courte s'avère nécessaire, la sélection peut se faire au moyen de l'entrée des codes correspondants ou par des touches fonctions dénommées de façon appropriée sur l'écran ou sur les touches elles-mêmes.

Les options du menu doivent être formulées pour permettre le pointage ou l'entrée d'un code plutôt qu'exprimées comme questions.

Si l'entrée se fait par des codes, chaque code devra être la(les) lettre(s) initiale(s) des labels des options, plutôt qu'un nombre (ou un code alphanumérique) arbitraire, excepté quand il y a un ordre implicite ou pour des listes très longues. Ces codes doivent être utilisés de façon homogène dans toutes les étapes du dialogue. Les labels auxquels ils correspondent doivent être localisés aussi de façon homogène.

S'il y a une numérotation, elle doit commencer par 1 et pas par 0. Après ce nombre, il doit y avoir un point et un espace, ainsi qu'un point après le texte de l'option. Ces nombre doivent être justifiés à droite.

Le texte des options doit être aligné en colonnes et justifié à gauche.

Les options d'un menu doivent être présentées selon un ordre logique du point de vue de la tâche, ou bien s'il n'existe pas un tel ordre, selon leur

fréquence d'utilisation. Quand les deux ordres précédents sont inopérants, il convient alors d'utiliser l'ordre alphabétique.

Les menus doivent être formatés de façon à refléter l'organisation hiérarchique des options, plutôt que sous la forme d'une liste amorphe.

Chaque menu affiché ne devrait comporter que les options appropriées à l'étape de la transaction dans laquelle se trouve l'utilisateur.

Quand la sélection doit se faire à partir d'une longue liste d'options, et que toutes les options ne peuvent être affichées sur un même écran, une séquence hiérarchique de menus doit être offerte plutôt qu'une longue liste de plusieurs pages.

Dans les menus hiérarchisés, l'accès à des options critiques et/ou fréquentes doit être facilité. De plus, la position dans la structure des menus doit être fournie (e.g., en présentant une liste des options successives déjà sélectionnées).

Le nombre d'étapes dans une suite de menus doit être minimisé.

Les menus doivent être présentés successivement à la même place, plutôt que simultanément à différents endroits de l'écran.

Le format des différents menus doit être homogène à chaque niveau de menu et une fonction d'accès au niveau immédiatement supérieur doit être fournie.

Toutes les options doivent être bien entendu cohérentes du point de vue de leur dénomination et de leur ordre.

Quand une sélection est faite, il doit y avoir un feed-back immédiat (e.g., message, illumination de l'option choisie).

Pour des utilisateurs plus expérimentés, il est souhaitable de fournir la possibilité d'utiliser des raccourcis, d'anticiper la séquence des options par l'emploi de commandes (lesquelles peuvent être les codes des options).

d - Touches fonctions

L'utilisateur indique les actions voulues en pressant des touches, chacune d'entre elles représentant une commande, un modificateur de commande ou une valeur de paramètre. C'est habituellement un type de dialogue à l'initiative de l'utilisateur, mais cela peut aussi être à l'initiative de l'ordinateur quand les touches fonctions sont programmables et quand il y a guidage sur l'écran.

Les touches fonctions comportent un certain nombre d'avantages. L'utilisateur n'a qu'à reconnaître la fonction (au lieu d'avoir à la rappeler), ce qui occasionne plus d'erreurs. Toutes les options sont toujours visibles et immédiatement disponibles. Les touches fonctions permettent assez facilement d'apprendre tout un ensemble d'entités, rendent la frappe aisée et évitent les erreurs typographiques. Pour réduire encore plus les efforts et le temps de transaction, on peut regrouper spatialement (et/ou avec codage couleur) les fonctions les plus fréquentes ou celles qui tendent à être utilisées conjointement dans une même séquence.

L'inconvénient majeur des touches fonctions est que lorsqu'elles sont nombreuses, cela nécessite des claviers volumineux, plus coûteux, et cela permet difficilement des changements d'applications informatiques. Par ailleurs il y a le risque de trop simplifier la tâche, mais surtout, même s'il y a moins de risques d'erreurs typographiques par rapport à la frappe de caractères, les erreurs de touches fonctions peuvent avoir des conséquences plus importantes.

Assigner plusieurs fonctions à la même touche, même si c'est une solution pour réduire l'espace du clavier, est générateur d'erreurs, pas seulement parce que l'utilisateur doit presser plusieurs touches à la fois, mais parce qu'il risque d'oublier dans quel mode fonction il se trouve. En général, une seule fonction doit être affectée par touche, en particulier pour un

ensemble assez réduit de fonctions de type routinier, ou pour un ensemble de fonctions distinguables aisément d'autres fonctions (e.g., édition d'écran et entrée de données).

Ce type de dialogue est particulièrement approprié pour des utilisateurs naïfs, quand la syntaxe est simple ou bien quand un guidage est effectué. Si ce n'est pas le cas, une formation est nécessaire.

Les touches fonctions sont particulièrement utiles quand la tâche requiert un nombre limité de commandes, en complément d'autres types de dialogue, pour entrer des données critiques, du point de vue de la tâche, quand cela doit être fait rapidement et sans erreur de syntaxe. Elles sont très utiles pour des entrées très fréquentes (e.g., fonctions de type validation, impression, changements de page, etc.), et pour des entrées "provisoires", i.e., des actions qui peuvent être faites avant la fin d'une transaction (e.g., fonctions de tabulation, de répétition, d'appel à des valeurs par défaut, help, etc.).

Les touches fonctions doivent être codées de façon informative pour désigner les fonctions qu'elles exécutent. Ces fonctions doivent être assignées de façon cohérente tout au cours du dialogue. Quand plusieurs fonctions sont assignées à une même touche, elles doivent être choisies de la façon la plus homogène possible (par exemple, une fonction appelée Arrêt ne doit pas dans un cas effacer les données, les sauver dans un autre, et signaler l'exécution dans une troisième). Dans tous les cas, il devrait y avoir un affichage indiquant quelle fonction est active.

Les fonctions qui ne sont pas nécessaires dans une transaction particulière devraient être désactivées, rendues indisponibles. Si l'utilisateur presse une telle touche, aucune action ne doit avoir lieu, si ce n'est un message d'aide. Quand certaines fonctions sont actives et d'autres pas, le sous-ensemble des fonctions actives devrait être indiqué (e.g., intensité lumineuse supérieure).

Les touches fonctions devraient être regroupées dans un endroit distinct et facile d'accès. Les touches correspondant à des fonctions d'urgence

doivent être disposées de façon prégnante, avec un codage distinctif (e.g. taille ou couleur).

Les touches ayant un effet potentiellement destructeur devraient être protégées physiquement.

Les touches fonctions doivent nécessiter une seule pression pour être activées. Si elles sont pressées à plusieurs reprises consécutivement, leur effet ou fonction ne doit pas changer pour autant.

L'exécution des fonctions doit toujours être signalée à l'utilisateur ou bien par une réponse immédiate et observable (e.g. son résultat), ou bien par un message, ou par l'illumination de la touche.

e - Langages de commande

Les langages de commande à l'initiative de l'utilisateur sont acceptables pour des utilisateurs entraînés qui ont intériorisé le modèle des fonctions du système et la syntaxe du langage. Comme ils satisfont à ces critères, ce type de dialogue est habituellement préféré et convient bien aux concepteurs de systèmes et aux programmeurs. Cependant, ces derniers fournissent souvent sans distinction de tels langages à des utilisateurs qui ne satisfont pas aux critères précédents. Dans ce cas, i.e., pour des utilisateurs non expérimentés, cela conduit à des erreurs, à des confusions et à des frustations.

Les langages de commandes sont indiqués pour les systèmes disposant de nombreuses fonctions, et pouvant accepter un large éventail d'entrées, notamment quand ces entrées comportent des séquences arbitraires.

L'avantage d'un tel type de dialogue par rapport à des touches fonctions est qu'il autorise plus de flexibilité pour changer les fonctions à mesure que le système évolue ou bien lorsqu'il apparaît que certaines fonctions vont être plus ou moins fréquemment utilisées. Les changements sont alors des modifications du logiciel et pas des changements du matériel.

Cependant ce type de dialogue comporte un certain nombre d'inconvénients.

En particulier, l'apprentissage et le rappel d'un ensemble des commandes est plus difficile que la reconnaissance de touches fonctions. Par ailleurs, les utilisateurs n'emploient parfois qu'un sous-ensemble limité des commandes disponibles, ce qui les conduits à avoir recours à des documents, listings pour déterminer ou revoir les commandes utilisées moins fréquemment.

La nécessité de frapper plusieurs touches comporte un risque plus élevé d'erreurs de frappe, bien que de telles erreurs puissent être de moindre conséquence que celles occasionnées par la pression sur une mauvaise touche fonction. Taper plusieurs touches est aussi évidemment moins rapide que de presser une seule touche fonction. Ces inconvénients sont d'autant plus évidents que les utilisateurs ne sont pas des dactylographes expérimentés. Cependant, on peut estimer que des individus ayant eu une expérience même limitée avec une machine à écrire et qui n'ont à entrer qu'un ensemble relativement restreint et bien défini de commandes peuvent apprendre à le faire dans un laps de temps assez court, de l'ordre de quelques semaines.

L'aire d'entrée de commandes doit être située de façon consistante sur chaque page d'écran, preférentiellement en bas de l'écran. A côté de cette aire, il doit y avoir une aire d'affichage pour le guidage des entrées et pour la récapitulation des séquences de commande entrées précédemment.

L'utilisateur doit pouvoir effectuer ses manipulations, lancer ses commandes sans aucune considération liée au traitement informatique interne. Par exemple, l'utilisateur doit être capable de demander un fichier par son nom, sans autre information supplémentaire, telle que la localisation physique du fichier.

Le nombre de commandes doit être aussi réduit que possible. Il faut éviter l'existence de plusieurs commandes pour la même fonction.

Un langage de commande doit être organisé en groupes de commandes, fonctionnellement similaires, et à différents niveaux de complexité pour tenir compte des niveaux d'expérience des utilisateurs.

L'utilisateur doit pouvoir requérir des aides pour déterminer la nature des paramètres de commande requis. Pour les utilisateurs naïfs, il peut être souhaitable d'avoir un guidage automatique.

Les commandes doivent pouvoir être entrées sans ponctuation. Si une ponctuation est nécessaire, alors un symbole standard devrait être utilisé de façon homogène.

L'utilisation de quantificateurs et de connecteurs logiques, d'emploi malaisé, doit être évitée.

L'utilisateur ne doit pas avoir à faire de distinction liée au nombre d'espaces blancs dans une expression de commande; une séquence de commandes doit être valide qu'il y ait un seul espace, deux ou plus, avant ou entre les commandes ou les paramètres.

Le logiciel devrait être capable de reconnaître les erreurs de frappe ou de syntaxe les plus communes et corriger les commandes avec confirmation de l'utilisateur plutôt que de requérir une nouvelle entrée.

Les séquences de commande doivent être dictées par les choix de l'utilisateur plutôt que par des considérations de traitement interne (e.g., un utilisateur devrait pouvoir entrer ses données dans l'ordre où elles lui parviennent).En général, un langage de commande est constitué d'un terme de commande, qui définit une fonction et souvent de paramètres qui spécifient les différentes options de la fonction. Le format d'une commande peut être de type positionnel (dans ce cas les paramètres sont identifiés par leur position fixée ou relative), ou bien de type mots-clefs (dans ce cas ce sont les mots-clefs qui identifient à la fois le type des paramètres et leur valeur). Le format positionnel, qui impose une plus grande charge mémoire, n'est pas recommandé pour des utilisateurs débutants.

Il peut arriver qu'une commande soit lancée avec un ou plusieurs paramètres manquants. Dans ce cas, le logiciel ne devrait jamais ignorer complètement la commande. Une possibilité est d'indiquer à l'utilisateur ce qui manque. Une meilleure solution est d'assigner automatiquement une valeur (par défaut) aux paramètres manquants. Ces valeurs par défaut sont les valeurs le plus souvent utilisées dans les mêmes circonstances. Bien évidemment, si des valeurs par défaut sont proposées, l'utilisateur doit en être informé, et pouvoir les confirmer ou les changer aisément.

Un certain nombre d'autres considérations qui s'appliquent aux langages de commande, mais pas exclusivement, seront évoquées dans la section sur le codage.

3 - Procédures de dialogue

Procédure signifie ici la façon d'utiliser un type d'entrée particulier, i.e., la séquence suivant laquelle les commandes, ou la sélection d'options de menu, etc. vont être passées. Quelques considérations peuvent aider à la conception de ces procédures.

D'abord, du fait que les utilisateurs naïfs ont tant à assimiler, les procédures doivent être conçues de manière à limiter leur charge en mémoire. Une bonne méthode est d'incorporer des aide-mémoire dans les procédures. Une façon de le faire est d'utiliser des langages de commande dans lesquels les commandes font effectivement bien ce qu'attendent les utilisateurs. Une autre est d'utiliser des valeurs par défaut pour les données ou paramètres que l'utilisateur omet. Une autre façon est de fournir des menus.

Par ailleurs, le concepteur devrait fournir des ressources informatiques qui permettent aux utilisateurs d'augmenter progressivement la complexité et la sophistication de leurs interactions. Cela signifie par exemple: permettre à l'utilisateur de supprimer certaines des procédures les plus banales (i.e., les plus fréquentes, et souvent les mieux connues) qui ont été fournies au bénéfice des naïfs. Cela signifie aussi permettre à l'utilisateur de développer des procédures définies selon ses besoins (e.g., macro-fonctions sur mesure).

Enfin, il est souhaitable de prévoir une introduction aux procédures de dialogue, pour les naïfs. Cela peut être un guidage on-line ou un document permettant à l'utilisateur d'effectuer une ou plusieurs transactions complètes avec le système afin d'accomplir un objectif. Ceci peut se faire au moyen de scenarios, i.e., de transactions enregistrées entre le système et des utilisateurs expérimentés, ce qui permet à un utilisateur naïf d'observer comment un expérimenté utilise les procédures existantes pour atteindre un objectif particulier. Une fois qu'un ensemble suffisant de procédures a été défini, les utilisateurs peuvent s'y essayer sur des exemples dont les solutions sont connues du logiciel.

Idéalement, de cette manière, le logiciel peut amener l'utilisateur vers la bonne solution, au pas à pas quand c'est nécessaire, et à mesure que l'utilisateur acquiert de l'expérience, le logiciel intervient de moins en moins.

On reviendra sur certains de ces aspects procéduraux dans la section sur les entrées, à propos des séquences de commande.

B - LES ENTREES

1 - Entrée d'informations

a - Mode d'entrée

De façon générale, sauf pour l'entrée de texte, l'entrée d'informations doit se faire par remplaçement direct de caractères (par exemple caractères de soulignement à des fins de guidage). L'entrée d'information par sur-frappe de caractères alphanumériques est à proscrire.

b - Changements de mode

Les passages d'un mode à un autre (e.g., mode shift/ mode normal; clavier/ souris) doivent être minimisés, à moins qu'ils ne correspondent à des distinctions explicites au sein de la tâche d'entrée.

c - Actions minimales

De façon générale, l'utilisateur ne devrait pas avoir à entrer les mêmes données plusieurs fois ou à entrer des données qui pourraient être dérivées par le logiciel d'autres données déjà entrées.

Quand des données sont utiles pour des transactions ultérieures, ces données devraient être automatiquement affichées.

Excepté pour les numéros de série, l'entrée de caractères "0" en début d'item doit être optionnelle.

L'utilisateur ne doit pas avoir à faire la distinction entre un ou plusieurs espaces blancs.

L'utilisation de délimiteurs spéciaux entre données doit être évitée.

Quand les items à entrer sont de longueur variable, l'utilisateur ne doit pas avoir à les justifier que ce soit à gauche ou à droite, ni à supprimer des caractères de délimitation inutilisés.

La double frappe, les codes contrôle doivent être évités.

Quand des codes arbitraires doivent être entrés, utiliser de préférence des menus.

d - Valeurs par défaut

Les valeurs courantes par défaut doivent apparaître sur l'écran, dans les champs d'entrée appropriés. L'acceptation ou le rejet de ces valeurs par l'utilisateur devrait se faire de façon simple (e.g., tabulation ou touche de confirmation pour acceptation et sur-frappe pour rejet).

e - Taille des entrées

Comme pour les commandes et les codes en général, la longueur des items ne devrait pas dépasser 5 à 7 caractères. Quand des informations de longueur supérieure doivent être entrées, elles devraient être découpées en items plus courts.

Le nombre de caractères à frapper doit être minimisé, par exemple en utilisant des abréviations signifiantes et distinctes. Les abréviations doivent suivre une règle explicite (e.g., troncature) et devraient être de longueur fixe. Les abréviations ne suivant pas la règle usuelle devraient être peu nombreuses et utilisées pour des exigences de clarté. Quand ces abréviations ne sont pas reconnues, un dialogue de clarification doit s'établir.

f - rythme et ordre

L'entrée de données doit se faire au rythme de l'utilisateur et non pas au rythme du logiciel.

Quand des parties de longues informations sont très familières, elles devraient être entrées les dernières (excepté si cela contredit la logique ou l'usage établi).

Quand l'entrée de données comprend la transcription de documents source, les écrans de remplissage doivent correspondre aux formulaires papier (c'est aussi souhaitable pour le mode question/ réponse en ce qui concerne l'ordre des données).

Quand aucun document papier n'est concerné, la séquence des entrées doit suivre un ordre logique, du point de vue de la tâche, auquel on sait que l'utilisateur s'attend.

g - Edition/correction

L'édition doit se faire de façon consistante, par remplacement direct des anciennes données par les nouvelles. Quand il y a changement de données,

les données anciennes et les nouvelles doivent apparaître pour confirmation par l'utilisateur.

h - Validation d'entrée

La cessation de l'entrée de données doit se faire de façon explicite (e.g., le retour au menu ne doit pas résulter dans le traitement des données entrées). La touche de validation ou fin d'entrée doit être dénommée explicitement.

Dans le remplissage de formulaires, l'entrée d'items logiquement reliés doit se faire par une action explicite à la fin plutôt qu'une entrée séparée pour chaque item. L'utilisateur doit pouvoir revoir les données entrées. Quand cela est possible, l'utilisateur devrait avoir la possibilité de modifier les données si nécessaire, avant l'action explicite d'entrée ou de validation.

I - Feed-back

Excepté pour les entrées de type confidentiel (e.g., mots de passe), toutes les entrées effectuées par l'utilisateur doivent apparaître sur l'écran.

j - Curseur

Quand la désignation de position se fait par le moyen d'un curseur, le curseur doit avoir des caractéristiques visuelles distinctes et ne doit pas obscurcir le caractère désigné. L'activation de la position doit se faire d'une façon explicite et distincte du placement du curseur. Un feed-back direct (clignotement ou changement d'intensité lumineuse) doit être fourni, avec un temps de réponse court.

S'il y a une position d'origine (home position), celle-ci doit être consistante sur tous les affichages.

Le curseur doit rester où il se trouve jusqu'à ce qu'il soit déplacé par l'utilisateur. Cependant, dans le cas de remplissage de formulaires, le curseur doit être placé automatiquement au premier caractère du premier champ d'entrée. De plus, les mouvements du curseur doivent être minimisés. Par

exemple, il faut aligner les champs d'entrée, prévoir une touche de tabulation permettant de passer d'un champ à un autre, grouper les entrées requises en haut de l'écran et celles qui sont optionnelles en bas, sauf quand cela contredit d'autres critères de regroupement (logique, séquentiel, etc.).

Les déplacements séquentiels entre champs devraient se faire plutôt avec des touches fonctions que de façon automatique.

Les mouvements du curseur doivent être consistants dans toutes les directions et isomorphes à la taille des caractères.

Les aires non nécessaires pour l'entrée de données doivent être protégées et le curseur ne devrait pas s'y arrêter.

k - Pointage direct

Le pointage direct doit être utilisé plutôt que le curseur quand la sélection d'items est l'objectif principal de la tâche.

Quand un utilisateur doit contrôler le déplacement d'objets, le déplacement de la main de l'utilisateur doit être identique à celui de la direction du mouvement de l'objet.

Les écrans tactiles sont recommandés pour les dialogues à base de menus.

La tablette graphique avec un stylet est recommandée pour les entrées graphiques. Cependant, la tablette graphique n'est pas un très bon outil pour l'entrée de caractères (la reconnaissance de caractères à la main est d'ailleurs envisageable seulement quand le dialogue n'a pas besoin d'être rapide).

La tablette graphique doit être au moins aussi grande que l'écran.

l - Entrée vocale

L 'entrée vocale peut être considérée pour l'identification des utilisateurs.

L'entrée vocale représente une option possible quand les autres canaux sont saturés.

L'entrée vocale , en alternance avec d'autres modes d'entrée, est utile pour distinguer par exemple entre données et commandes (e.g., commandes par la voix et données par clavier).

L'entrée vocale ne doit pas être utilisée dans un environnement sonore élevé.

Les appareils de reconnaissance doivent être "entraînés" dans des conditions qui reproduisent de façon aussi proche que possible les conditions d'utilisation.

La répétition des mots pendant l'entraînement doit être randomisée. Les vocabulaires utilisés doivent être tels que les mots et les phrases enregistrés soient aussi dissemblables que possible.

Le nombre de mots admis doit être limité.

Le microphone d'entrée doit comporter un bouton de déconnexion pour éviter des confusions avec les conversations annexes.

m - Guidage

Certains principes sur le codage et le labeling (qui seront détaillés plus loin) s'apparentent au guidage. Par exemple:

L'entrée de données doit être guidée explicitement par des messages d'aide et des labels de champs, explicites et placés de façon consistante sur les écrans par rapport aux aires d'entrée (e.g., toujours au dessus à gauche).

Ces labels doivent être dénommés de façon distinctive, afin d'éviter des confusions avec d'autres catégories (entrée de données, commandes, etc.).

Quand l'écran est encombré (ce qui ne devrait pas arriver), un code auxiliaire doit être utilisé (inverse-video, intensité lumineuse, couleur) pour distinguer les éléments affichés, en particulier pour distinguer labels et données.

Seuls des termes sur lesquels les utilisateurs s'accordent doivent être utilisés.

Les labels et champs d'entrée doivent être compatibles avec les labels et champs d'affichage.

Le label de chaque champ d'entrée devrait se terminer par un symbole spécial (e.g., ":") signifiant qu'une entrée est attendue. Des éléments additionnels de guidage peuvent aussi être utiles (e.g., Date (JJ/MM/AA: _ _/_ _/ _ _)). D'autres guidages implicites permettent par exemple d'indiquer une longueur fixe ou maximale d'entrée, et permettent aussi de distinguer entre entrées requises (e.g., soulignement par tirets) et entrées optionnelles (e.g., soulignement par points).

Quand des unités de mesure sont associées avec une entrée, elles devraient apparaître dans le label du champ d'entrée au lieu d'avoir à être entrées par l'utilisateur. Bien entendu, ces unités de mesure doivent être familières aux utilisateurs et ne pas requérir un traitement des données brutes de la part de ces derniers.

Par ailleurs, les labels doivent être protégés des interventions des utilisateurs, à moins que des changements soient permis par l'utilisation explicite d'une fonction d'édition des labels. Quand plusieurs items (particulièrement s'ils sont de longueur variable) doivent être entrés, il doit toujours y avoir un espace entre les champs d'entrée. Cet espace doit être protégé et un signal auditif devrait alerter l'utilisateur quand celui-ci tente d'entrer dans ce champ.

n - Détection d'erreurs

Le logiciel devrait vérifier les entrées afin de détecter les erreurs de format ou de contenu et quand des entrées ont été différées. Dans tous les cas, l'utilisateur doit être informé de ses erreurs ou des données manquantes et devrait être autorisé à effectuer des corrections avant qu'une autre transaction ne débute.

2 - Dénomination

a - Généralités

S'appliquant aussi bien aux termes de commande qu'aux labels des touches fonctions ou aux options de menus, l'activité de dénomination est cruciale pour la qualité de l'interface homme-ordinateur.

Pour rendre la communication homme-ordinateur aussi naturelle que possible, i.e., prédictible par l'utilisateur, les entrées qu'il effectue doivent appartenir à un langage correspondant bien à ses attentes, capacités et expérience.

Un langage tel que le langage naturel est certainement le plus familier. Cependant ce n'est pas toujours le bon choix. Dans de nombreuses applications spécialisées, d'autres langages tel le langage mathématique, sont probablement des formes de communication meilleures que des termes français.

La langue naturelle n'est un moyen utile de communication que dans des situations où il y a peu de jargon, où les messages n'ont pas à être répétés souvent, et quand l'idée à transmettre est définie précisément.

Dans les communications avec l'ordinateur, il y aura toujours des limitations sémantiques (i.e., liées au sens) ou syntaxiques (i.e., liées à l'arrangement des mots selon des règles) dans l'utilisation du langage naturel. De plus, les utilisateurs n'accepteront un langage restreint que s'il est adéquat pour décrire les tâches à exécuter.

Contrairement aux langages qui ont évolué au cours du temps (à part peut être l'espéranto, qui n'est d'ailleurs pas une vraie langue naturelle), un langage de commande est une création du concepteur. Bien qu'un langage de commande soit restreint, le concepteur peut créer un langage qui n'a pas les inconvénients du langage naturel. Le concepteur peut définir un vocabulaire très précis, mais ce vocabulaire est forcément en interaction avec le vocabulaire du langage naturel. Un nouveau langage de commande et le langage naturel doivent bien s'harmoniser. Des incohérences conduisent à une performance dégradée de l'utilisateur. Souvent, les concepteurs tendent à croire qu'il existe une correspondance univoque entre les termes de commande et leur signification, i.e., les fonctions auxquelles elles correspondent, alors que souvent les utilisateurs identifient certaines commandes comme étant identiques. Par ailleurs, les concepteurs imaginent que les utilisateurs ne font aucune hypothèse, et s'en tiennent strictement à ce qui leur est dit sur la commande, alors qu'on sait qu'ils effectuent de nombreux présupposés. Enfin, les concepteurs croient aussi que les déductions, interprétations faites par les opérateurs se font à partir d'un système de référence invariant, alors que la signification d'une commande est très dépendante du contexte immédiat.

b - Recommandations

Avec toutes les réserves liées aux tâches et applications particulières, il est possible de fournir quelques recommandations de conception des langages de commande: un bon langage de commande doit être précis, homogène, de taille raisonnable, flexible, permissif, et comporter un guidage.

b1 - Précis

Tout d'abord, la dénomination des commandes devrait refléter le point de vue de l'utilisateur plutôt que celui du programmeur. Des termes signifiants doivent être employés; par exemple, il vaut mieux utiliser "ajouter", "tracer", "sortir" que "option1", "option2", "option3". Des termes communément admis doivent être utilisés (e.g., o pour oui et n pour non au lieu de 0 ou1).

Un langage de commande doit être simple et le comportement du logiciel doit être prédictible et apparaître consistant logiquement en toutes circonstances.

Une commande doit décrire exactement la fonction a exécuter. Par exemple, imprimer pourra signifier imprimer une ligne, ou une page, ou un texte, alors que lister pourra signifier lister le contenu d'un fichier.

L'utilisation de termes sémantiquement proches, tels Ajouter, Somme et Total doit être bannie.

b2 - Homogène

Tous les termes de commande doivent être utilisés de façon homogène et standardisés du point de vue du sens, d'une transaction à une autre, et souligner les différences de fonction (e.g., deux commandes ne devraient pas être dénommées respectivement "Afficher" et "Voir" quand l'une permet l'édition et l'autre pas). Les termes doivent aussi être standardisés. Par exemple, si 1 représente oui et 0 représente non, alors cette représentation doit être maintenue tout au cours du dialogue (bien que y et n eussent été préférables). De plus, cette représentation ne doit pas être en conflit avec les conventions habituelles, comme 4 pour oui et 6 pour non.

b3 - Flexible

Il devrait être possible aux utilisateurs de dénommer eux-mêmes leurs fichiers ou commandes, notamment ceux et celles qui sont utilisés fréquemment.

L'utilisateur doit pouvoir sentir qu'il contrôle le dialogue en entrant ses commandes et, à mesure qu'il acquiert de l'expérience, il doit avoir l'option d'utiliser des formes plus brèves de commande.

b4 -Taille

Bien que le nombre de commandes dans une séquence ne doive pas être trop élevé, l'utilisation de codes, de commandes mnémoniques et de

formats fixes doit etre limitée. D'un autre côté, le concepteur ne peut donner entière liberté d'utilisation du langage naturel. Dans le premier cas, les commandes seront trop structurées, dans l'autre pas assez.

Chaque entrée doit être courte afin de favoriser la détection d'erreurs.

b5 - Permissif

Le logiciel devrait permettre des erreurs, en particulier des erreurs de frappe ou des fautes d'ortographe.

Le logiciel devrait être capable de distinguer des abréviations proches et de reconnaître par exemple "i", ou "imp" pour imprimer.

Le logiciel doit être capable d'interpréter capitales ou minuscules comme ayant la même valeur et reconnaître les erreurs de frappe comme l'utilisation inappropriée du shift (e.g., 1 et & sur un clavier azerty, ou 4 et $ sur un qwerty).

Entrer une commande et ses paramètres ne devrait pas requérir l'utilisation contraignante de zéros ou de blancs.

Quand l'entrée de paramètres est nécessaire, l'utilisateur doit pouvoir les entrer individuellement, dans une séquence, ou pas du tout (en utilisant des valeurs par défaut), selon son degré d'expérience.

b6 - Guidage

Il doit y avoir une forme de guidage à la demande, aisément accessible dans le cas où l'utilisateur a oublié certaines commandes. De plus, à chaque étape des séquences d'action, l'utilisateur devrait pouvoir demander ce qui est attendu de lui, ce qui lui est possible.

Dans l'avenir, le logiciel pourra peut-être fournir une aide automatique chaque fois qu'il pourra déterminer que l'utilisateur est en difficulté.

c - Abréviations

Chaque fois que possible, le concepteur devrait fournir pour les utilisateurs expérimentés un ensemble d'abréviations bien choisies, en particulier pour les commandes répétitives (dans le cas de commandes répétitives, on peut envisager l'utilisation de touches fonctions, dans une aire specialisée).

Dans l'utilisation d'abréviations, il convient de n'employer que celles qui sont le plus largement admises car les utilisateurs pourront avoir rencontré certaines abréviations qui signifient autre chose. Par exemple, "e" peut signifier éditer sur un système et effaçer sur un autre.

L'utilisateur doit pouvoir entrer à sa guise une abréviation ou la commande complète.

Les abréviations sont déconseillées pour les affichages.

Les utilisateurs doivent être informés de la règle d'abréviation utilisée.

Ils doivent pouvoir définir leurs abréviations eux-mêmes.

Il ne doit pas y avoir de ponctuation dans une abréviation.

La troncature est une méthode d'abréviation recommandée.

Les abréviations doivent être beaucoup plus courtes que la commande complète.

Il ne doit y avoir qu'une abréviation par commande.

L'auto-complètement des abréviations est deconseillé.

3 - Séquences de commande

Tout en rappelant que la conception des transactions doit être établie en fonction d'analyses du travail préalables, et représenter des unités logiques du point de vue de l'utilisateur, on peut définir un certain nombre de recommandations concernant les séquences de commande (ou séquences de fonctions, par opposition à entrée de données).

a - Simplicité

Les séquences de commande doivent être simplifiées au maximum, en particulier pour des tâches en temps réel qui requièrent des actions rapides de la part de l'utilisateur.

b - Conformité des buts

La facilité des séquences de fonctions doit correspondre aux buts poursuivis. Par exemple, les actions fréquentes doivent être faciles à exécuter alors que les actions potentiellement destructrices doivent être suffisamment difficiles pour nécessiter une certaine attention de la part de l'utilisateur.

c - Conformité vis-à-vis de l'expérience

Les séquences de commande doivent aussi correspondre aux capacités des utilisateurs: pas-à-pas pour les inexpérimentés, transactions rapides et puissantes pour les expérimentés (e.g., raccourcis, anticipation par empilage de commandes). Il doit être possible aux utilisateurs de définir leurs propres macro-commandes pour les séquences de commande fréquemment utilisées.

d - Contrôle

Les séquences de commande doivent être contrôlées par l'utilisateur, en fonction de ses besoins, disponibilités, attention, plutôt que par le traitement interne ou les événements extérieurs.

L'ordinateur ne doit pas interrompre l'utilisateur en cours d'entrée pour requérir des corrections mais attendre jusqu'à ce que l'utilisateur signale la fin de la transaction (ou demande de l'aide).

Idéalement, la plupart des actions de l'utilisateur devraient être anticipées et des options appropriées à chaque cas devraient être fournies. Ceci permet à l'utilisateur d'avoir toujours la main. Il ne doit jamais y avoir de voie sans issue dans une transaction, sans aucune action ultérieure possible.

e - Caractère explicite

Les séquences de commande doivent résulter d'actions explicites de la part de l'utilisateur et non pas être une conséquence de traitements antérieurs.

f - Entrées différées

Si les entrées de l'utilisateur doivent être différées, le clavier (ou autre moyen d'interaction) doit être bloqué jusqu'à ce que la prochaine transaction soit possible. Ceci doit être accompagné par la disparition du curseur ou par un signal auditif. Quand la transaction peut reprendre, le logiciel doit en informer l'utilisateur.

g - Cohérence et contexte

Les séquences de commande doivent être consistantes du point de vue de leur forme et de leurs conséquences: des moyens similaires doivent être mis en oeuvre pour des résultats similaires, d'une transaction à une autre, et d'une tâche à une autre. Si les conséquences d'une commande diffèrent selon le contexte (établi à partir d'une commande passée précédemment), un moyen approprié d'affichage du contexte doit être fourni à l'avance à l'utilisateur. Rappelons aussi que les affichages doivent être conçus de sorte que les éléments pertinents de l'entrée de commandes (listes d'options, aires d'entrée, etc.) soient distincts en position et en format.

h - Interruptions de dialogue

L'utilisateur doit pouvoir interrompre, différer ou annuler de façon consistante une transaction en cours. Si disponibles, les fonctions suivantes peuvent être définies.

Une option d'annulation doit avoir l'effet de régénérer l'écran, sans traiter aucun des changements effectués par l'utilisateur.

Une option de retour-arrière doit avoir l'effet de revenir à l'affichage obtenu dans la transaction précédente.

Une option de recommencer (restart) doit avoir l'effet de retourner au premier affichage d'une séquence de transactions, afin de permettre à l'utilisateur de revoir sa séquence d'entrées et d'effectuer les changements nécessaires.

Une option détruire doit avoir l'effet d'annuler toutes les entrées dans une séquence de transactions. L'utilisateur, en particulier pour une activité d'entrée de données, doit pouvoir confirmer son ordre d'annulation.

Une option de fin doit avoir pour effet de conclure une séquence de transactions et de retourner au menu général d'options.

i - Feed-back

Toutes les commandes doivent être répercutées de façon non ambiguë soit par leur exécution immédiate, soit par un message indiquant que l'exécution est en cours ou différée, ou que les commandes nécessitent une correction ou une confirmation. Le feed-back pour les entrées de commande devrait consister en des changements d'état ou de valeur des éléments affichés affectés par les fonctions considérées, de façon attendue et naturelle.

Aussi souvent que possible, le logiciel doit indiquer les options disponibles, celles qui sont actives et récapituler les options précédentes, à la demande de l'utilisateur.

Le logiciel doit garder trace du contexte de sorte que l'utilisateur n'ait pas à entrer à nouveau des commandes, des données ou des paramètres concernant la transaction en cours.

Le feed-back doit être consistant et distinct en position et dénomination.

J - Limitation des erreurs

Le logiciel doit d'une part ne pas provoquer d'erreurs et d'autre part anticiper l'erreur humaine. Par exemple, si l'utilisateur sélectionne une touche fonction invalide, aucune action ne devrait en résulter si ce n'est un message indiquant quelles fonctions sont appropriées à cette étape de la transaction.

L'utilisateur doit pouvoir modifier ses commandes avant son action explicite de validation.

Si une commande ou un paramètre n'est pas reconnu ou est inapproprié, il devrait y avoir un message indiquant de corriger cet élément, sans avoir à entrer à nouveau la commande complète pour confirmation par l'utilisateur.

Quand l'utilisateur entre une commande qui peut être destructrice ou causer des dommages importants aux données stockées, cette commande doit être explicitement confirmée. Le guidage pour une telle confirmation doit être clairement exprimé et la touche de confirmation doit être libellée de façon explicite et différente de la touche de validation.

Quand l'utilisateur termine une session (log-off) et s'il y a un risque de perdre des données, il doit y avoir un message le signalant et demandant confirmation de l'arrêt.

C - LES SORTIES

1 - Le temps de réponse

Le temps de réponse de l'ordinateur est le temps qui s'écoule entre une action (e.g., une commande) de l'utilisateur et une réponse de l'ordinateur.

Tout d'abord, il faut indiquer qu'il n'existe pas de temps de réponse unique acceptable car cela dépend de la tâche et des attentes de l'utilisateur. Cela signifie qu'un temps de réponse particulier peut être acceptable pour un utilisateur avec un certain objectif en tête, mais que ce ne sera pas acceptable pour le même utilisateur avec un autre objectif.

Cependant, bien que les capacités informatiques limitent le temps de réponse, un effort doit être fait pour garder le temps de réponse dans des limites acceptables. Ceci est important dans la mesure ou le temps de réponse affecte l'attitude de l'utilisateur vis à vis du système, ses habitudes de travail, et même les circonstances dans lesquelles il va utiliser l'ordinateur.

Un temps de réponse qui dépasse 10-15 secondes est totalement inacceptable car la continuité des processus de pensée de l'utilisateur devient difficile à maintenir. Quand le temps de réponse est aussi long, il faut restructurer la tâche.

Dans la plupart des applications informatiques, ce n'est pas un seul, mais des temps de réponse acceptables qui doivent être envisagés. En effet, les utilisateurs sont prêts à attendre de façon variable selon leurs types de requêtes. Par exemple, un utilisateur acceptera d'attendre plus longtemps à la fin d'une série de commandes plutôt que durant une seule transaction.

Un problème crucial concerne la variabilité du temps de réponse: celui-ci devrait être contrôlé de sorte que des délais identiques s'écoulent entre des actions identiques de l'utilisateur et la réponse de l'ordinateur.

De plus, chaque fois que le temps de réponse prévu est relativement long (i.e., plus de 5 secondes après le feed-back d'entrée de l'utilisateur), un

message devrait être affiché et indiquer que le logiciel est en cours de traitement. Si cela continue, ce message devrait apparaître toutes les 10 secondes.

Pour cependant donner un temps de réponse souhaitable, on s'accorde sur 2 secondes, en particulier pour la plupart des tâches de bureau, sauf si les transactions nécessitent l'accès à un ordinateur central, à un réseau.

Une dernière remarque est que des temps de réponse assez longs peuvent être acceptables quand les transactions sont généralement génératrices d'erreur (ce qui ne devrait pas être le cas), car des délais longs augmentent les possibilités d'auto-détection des erreurs préalablement à l'entrée.

Rappelons enfin que même si le temps de réponse est un aspect important de l'interface, il est encore plus important de s'assurer que l'information présentée est pertinente pour l'utilisateur.

2 - Etapes de conception des écrans

On peut distinguer deux étapes importantes dans la conception des écrans. La première concerne les caractéristiques de contenu, i.e., ce qu'il faut afficher. La seconde concerne les caractéristiques de surface des affichages, i.e., comment afficher.

Comme il a été indiqué précédemment, une activité des plus importantes dans la conception des écrans (mais aussi des entrées) est l'analyse de la tâche et de la population des utilisateurs. Bien que souvent ce soit l'analyse de l'activité qui reçoive le moins d'attention, c'est celle qui permet de savoir précisément ce qui doit être sur l'écran, à quel moment et en conséquence le type d'affichage à utiliser, sa taille, sa compatibilité avec d'autres affichages, etc.

Sachant cela, et en admettant que les analyses préalables ont permis de s'assurer que l'on possède les informations adéquates et de bons critères de

sélection, on peut suggérer quelques étapes importantes dans la conception des écrans:

- en fonction des données recueillies, sélectionner les informations à afficher, et leur place dans les séquences de transactions;

- choisir les moyens de présentation à employer (papier, pages écran, fenêtres, etc.);

- établir les groupements appropriés, du plus élémentaire (groupement d'items) au plus vaste (groupes de groupes), en fonction de principes explicites de groupement;

- développer des labels et des titres pour les items, les groupes;

- essayer différentes organisations spatiales, en utilisant des critères explicites et choisir celle qui permet la meilleure performance;

- choisir des titres pour les affichages;

- développer des instructions et autres guidages qui doivent être placés sur les écrans ou sur des supports annexes;

- tester les écrans et autres présentations sur des sujets représentatifs de la population des utilisateurs futurs;

- reconcevoir si nécessaire.

3 - Codage

Dans cette partie seront évoquées des généralités sur les caractéristiques souhaitables des codages, et des notions sur diverses formes de codage d'écrans.

Le codage s'utilise pour identifier des items particuliers (e.g., des commandes ou des labels d'écrans) et pour distinguer différentes classes

d'items simultanément présents (e.g., sur l'écran pour des informations, ou sur le clavier pour des touches fonctions). Les deux problèmes à considérer sont donc des problèmes d'identification et des problèmes de discrimination. Sans établir une frontière stricte entre ces deux problèmes, disons cependant qu'on traitera surtout de problèmes de discrimination dans la partie codage d'écrans et de problèmes d'identification dans une partie ultérieure sur le labeling (laquelle est en relation avec les considérations présentées précédemment sur la dénomination).

Dans tous les cas, il s'agit d'optimiser le choix du codage, c'est-à-dire d'une part de permettre une identification, une reconstruction et une compréhension rapides et aisées de la part de l'utilisateur et d'autre part d'éviter au maximum des confusions.

a - Généralités

Le codage doit être conçu en fonction des caractéristiques humaines, non pas pour s'adapter à des formulaires ou au logiciel. En d'autres termes, le choix d'un codage ne doit pas résulter de choix de codes liés à l'activité de programmation, ni se faire forcément à partir de codes existants. L'un comme l'autre de ces types de codes ont un sens, mais pas nécessairement pour l'utilisateur. Par exemple l'un peut résulter de contraintes d'organisation matricielle des items pour le programmeur; l'autre peut correspondre à des habitudes administratives, éventuellement obsolètes. Chacun d'entre eux ne correspond donc pas forcément aux caractéristiques de l'utilisateur considéré, pour la tâche qu'il se donne.

Dans la construction de codes, afin d'éviter les erreurs des utilisateurs, un certain nombre de règles devraient être suivies.

a1 - Unicité

Chaque code doit être unique (éviter toute proximité visuelle ou auditive entre codes). Comme les utilisateurs tendent à faire des erreurs sur des codes tirés de registres auditifs ou visuels similaires (i.e., qui sonnent de façon

similaire ou qui ont l'air similaire visuellement), il est souhaitable de choisir des codes tirés de vocabulaires distincts.

a2 - Familiarité

Les codes doivent être familiers chaque fois que possible, correspondre aux attentes et à l'expérience des utilisateurs, et clairement être associés avec les objets codés. Cependant, dans certains cas d'ambiguités avec le langage naturel, il peut être utile d'adopter un langage technique approprié.

La similarité d'un code vis-à-vis du descripteur complet d'un objet, d'une action, d'une instruction influence la compréhension. Les codes qui sont similaires à leur définition (i.e., qui contiennent les mêmes caractères et dans le même ordre), telle qu'elle est fournie à l'utilisateur (manuels, help, etc.) sont mieux compris et rappelés que ceux qui apparaissent différemment dans la définition.

a3 - Règle d'encodage

Utiliser autant que possible une règle systématique d'encodage, c'est-à-dire utiliser la même technique pour coder tous les items (sauf quand cela est en contradiction avec d'autres règles). La règle d'encodage utilisée devrait être connue (et donc explicitée) des utilisateurs. Les codes qui sont dérivés, de façon consistante, de règles établies ont en effet plus de sens que ceux dont les règles d'encodage sont soit inconsistantes, soit peu claires, car l'utilisateur, dans le doute, tend à utiliser la règle d'encodage connue.

a4 - Taille

Les utilisateurs ont des limites en termes de volume maximal d'informations perceptibles à un instant donné. Grouper de longs items ou codes en chunks de 5 caractères aide à compenser ces limitations (c'est le fameux chiffre magique 7 +/-2).

De façon générale, quand cela est possible, des codes courts (2 ou 3 caractères) sont préférables à des codes longs. Plus le code est long, plus le risque d'erreur est grand.

a5 - Conformité vis-à-vis du langage naturel

Les codes doivent avoir des similarités avec la langue française. L'ordre de similarité selon ce critère est le suivant: mots complets, abréviations standard ou usuelles, abréviations proches visuellement du mot ou celles qui sont facilement prononçables.

Les codes ne doivent pas comporter de combinaisons de lettres inexistantes ou inhabituelles dans le langage naturel de référence Par exemple, un code du type QVIL sera plus sujet à erreur que QUIL, car il n'y a pas de pattern du langage tel que QV (en tout cas en français).

Les codes doivent être prononçables car mieux répétables, mieux appris, et mieux rappelés que ceux qui sont difficilement prononçables.

a6 - Fréquence

La fréquence selon laquelle un code sera utilisé influence sa compréhension. Ainsi quand le caractère signifiant ne peut être fourni à tous les codes, quand on ne peut les optimiser tous, alors, ceux qui seront le plus souvent utilisés doivent avoir la priorité (notamment pour des premières utilisations d'un système particulier).

a7 - Contexte

Le contexte dans lequel un code est utilisé influence sa compréhension. En effet, les utilisateurs rappellent d'autant mieux des codes que le contexte au moment du rappel est similaire à celui de l'apprentissage. Par exemple, les codes qui sont placés avec consistance au même endroit, donneront lieu à une meilleure compréhension et un meilleur rappel que ceux apparaissant de façon désordonnée ou aléatoire, car la localisation fournit une indication complémentaire sur la signification du code Les codes qui font partie d'une longue phrase ont une meilleure signification que ceux qui se trouvent seuls, car la signification d'un code au sein d'un ensemble (set) est en partie acquise du set entier.

a8 - Codes mixtes

Les codes alphabétiques sont meilleurs que les codes numériques ou symboliques. Les codes tirés seulement de 2 vocabulaires (alphabétique et numérique) sont meilleurs que des codes tirés de 3 vocabulaires (e.g., alphabétique, numérique et symbolique).

a9 - Utilisation immédiate

Les codes, surtout s'ils sont nouveaux ou peu évocateurs, devraient être utilisés aussi immédiatement que possible après avoir été vus ou entendus.

a10 - Performance/préférences

Le choix des codes doit se faire en fonction de la performance des utilisateurs plutôt qu'à partir de leurs préférences. En effet, dans ce domaine, les opinions ne sont pas fiables.

b - Différentes formes de codage

Les éléments importants affichés sur l'écran, ou qui demandent une attention particulière de la part de l'utilisateur doivent être mis en valeur sur l'écran avec un codage auxiliaire, particulièrement quand ils apparaissent peu souvent (e.g., valeurs de variables récemment modifiées, données qui ne cadrent pas avec les valeurs attendues, valeurs dépassant les limites acceptables). Un codage positionnel peut suffire, mais des éléments très importants peuvent nécessiter d'autres formes de codage, telles que la couleur ou l'intensité lumineuse.

Par ailleurs, le codage d'écran doit être utilisé quand les éléments d'information sont disposés de façon irrégulière sur l'écran.

b1 - Alphanumérique

Les caractères alphanumériques peuvent être utilisés sans limite de combinaisons quand la présentation de données n'est pas elle-même alphanumérique (e.g., graphique).

Une convention consistante doit être adoptée pour choisir si les lettres seront soit en majuscules soit en minuscules. Pour les affichages, les lettres capitales sont plus lisibles. Pour les entrées, l'utilisateur doit pouvoir entrer indifféremment les deux.

Quand des codes combinent des lettres et des chiffres, les caractères de chaque type doivent être regoupés plutôt que dispersés (e.g., HW5 est meilleur que H5W).

Des codes signifiants doivent être adoptés plutôt que des codes arbitraires (e.g., DOS est meilleur pour dossier qu'un code numérique à 3 chiffres, en admettant que dossier est un bon terme pour l'objet considéré).

Les affichages (et les entrées) doivent être conformes aux stéréotypes de la population et aux abréviations acceptées.

Quand des codes ont une signification particulière (i.e., différente de leur acception usuelle), une définition doit être affichée en bas de l'écran (e.g., légende sur une carte). De plus, la définition doit répéter le code (e.g., Rouge=dangereux, en rouge).

Quand ils sont arbitraires, les codes qui doivent être rappelés ne doivent pas dépasser 4 ou 5 caractères (quand ils sont signifiants, comme un mot ou une abréviation, ils peuvent être plus longs).

Il faut éviter d'associer des paires de caractères qui risquent facilement d'être confondus, tels l-1, o-0, z-2.

b2 - Symbolique

Les symboles et autres codes doivent avoir une signification homogène d'un écran à un autre.

Sur des écrans alphanumériques, les symboles spéciaux, tels qu'astérisques, flèches, etc., peuvent être utilisés pour attirer l'attention de l'utilisateur sur des items particuliers. De tels symboles, s'ils sont utilisés pour des objectifs d'alerte, ne doivent pas être utilisés aussi pour autre chose.

Quand un symbole spécial est utilisé pour marquer un mot, il doit être séparé de ce mot d'un espace.

b3 - Couleur

Le codage couleur peut être considéré dans des applications où l'utilisateur doit distinguer rapidement parmi plusieurs catégories d'informations, particulièrement quand ces informations sont dispersées sur l'écran. Le codage couleur doit être utilisé de façon parcimonieuse, avec relativement peu de couleurs pour désigner des catégories d'informations critiques (il vaut mieux ne pas en utiliser plus de 5 ou 6).

Le codage couleur est une aide supplémentaire sur les écrans qui ont déjà été formatés aussi efficacement que possible avec une seule couleur. Il ne faut pas utiliser la couleur pour compenser un mauvais format d'écran. Il vaut mieux re-concevoir l'écran.

Le codage couleur peut convenir pour des titres, des données dépassant leur fourchette de valeurs normales, des données entrées récemment, des informations requérant une attention immédiate, pour souligner des champs importants, pour différencier des groupes d'informations, etc.

Le codage couleur doit être redondant avec d'autres caractéristiques, telle la symbologie. Chaque couleur ne doit représenter qu'une seule catégorie d'informations.

Ce codage doit être compatible avec les associations familières (e.g., nombres négatifs en rouge en comptabilité, rouge pour danger, jaune pour précaution, vert pour OK, bleu seulement pour des éléments de fond et pas pour des données critiques).

b4 - Autres codages (Intensité/Inverse vidéo/Clignotement/Son)

- Les différences d'intensité doivent seulement prendre 2 valeurs. Ce moyen peut être utilisé par exemple pour distinguer les labels de champs et les données en diminuant l'intensité des uns et en augmentant l'intensité des autres, ou pour souligner des options dans une liste, etc. Pas plus de 10% des éléments de l'écran doivent comporter une intensité lumineuse élevée.

- L'inverse vidéo peut être utilisée pour souligner des items qui nécessitent une attention particulière.

- Le codage par clignotement doit être utilisé en cas de besoin urgent d'attirer l'attention de l'utilisateur. Le clignotement devrait fonctionner sur un pointeur (e.g., une astérisque) plutôt que sur l'élément lui-même. L'utilisateur doit être capable d'arrêter le clignotement, lequel doit par ailleurs s'arrêter automatiquement dès que l'utilisateur a répondu. Même si on peut distinguer plusieurs fréquences de clignotement, il vaut mieux ne l'utiliser que comme codage binaire.

- Des sons distinctifs doivent être utilisés pour coder les items requérant une attention spéciale. Une variété de signaux de différentes fréquences est disponible. Les sons peuvent être présentés en séquence pour élargir le répertoire des signaux.

4 - Recommandations d'affichage

a - Caractéristiques générales des écrans

On peut identifier certaines caractéristiques ergonomiques souhaitables pour les écrans.

- Ils doivent être visibles, i.e., les éléments qui les composent doivent pouvoir être perçus par l'oeil humain. Il est donc nécessaire de s'assurer de la taille des caractères, de la fréquence de papillotement (flicker rate), du contraste, etc.

- Ils doivent être lisibles, i.e., communiquer de façon appropriée l'information nécessaire à l'utilisateur. S'assurer de la lisibilité signifie optimiser le groupage, l'espacement, l'ordre séquentiel des informations, etc.

- Ils doivent être consistants, cohérents entre eux vis-à-vis des séries de transactions et être compatibles avec les séquences d'entrée.

- Ils doivent être concis, i.e., fournir toute l'information nécessaire, pas plus, pas moins.

- Ils doivent fournir des informations directement utilisables, ne nécessitant qu'un minimum d'interprétations, de traductions, et de traitements de la part de l'utilisateur (quand ceux-ci peuvent être effectués automatiquement par le logiciel).

En plus de ces objectifs généraux de conception, certains domaines demandent une attention particulière. Il s'agit des messages, du contenu des écrans, du labeling, du groupement d'informations, de la standardisation, du texte, des listes, des tables, du défilement et des fenêtres, du rafraîchissement d'écrans.

b - Messages

Les messages peuvent comprendre une large palette d'informations. Ils peuvent être des suggestions (prompts) pour des entrées supplémentaires, des diagnostics d'erreur, des informations sur l'objet du travail, etc.

Quel qu'en soit le type, les messages doivent être concis et clairement compréhensibles par leur destinataire. Des codes sybillins, des acronymes, et des abréviations sont souvent difficiles à comprendre pour des utilisateurs débutants. Cependant, des utilisateurs plus expérimentés ne voudront pas de messages longs et pourront utiliser sans difficulté un ensemble d'abréviations bien construites. Malgré tout, dans ce dernier cas, des messages détaillés devraient être disponibles à la demande, même pour des expérimentés.

L'utilisateur qui est en période d'apprentissage ou l'utilisateur occasionnel tend à avoir besoin de plus d'instructions et d'aide de la part du logiciel. D'un autre côté, l'expert ou l'utilisateur fréquent ne veut rien voir apparaître qui ne soit essentiel pour atteindre le but qu'il s'est fixé. L'expérimenté souhaite des messages aussi courts et concis que possible et pourra s'estimer frustré s'il est obligé d'attendre que des messages contenant des informations inutiles ou redondantes défilent sur l'écran pour enfin accéder à ce qui l'intéresse. Les expérimentés devraient donc avoir la possibilité de supprimer ou d'interrompre des messages trop longs.

Une autre approche possible, comme indiqué précédemment, est de construire le logiciel à deux niveaux. Au premier niveau, le logiciel affiche des messages destinés au débutant; au second niveau, le logiciel présente un autre ensemble de messages destinés aux expérimentés. L'interaction devrait donc idéalement varier en fonction des capacités et du niveau de l'utilisateur.

Considérant que les utilisateurs peuvent être à mi-chemin entre ces deux catégories, d'autres facilités peuvent être incorporées au dialogue, comme par exemple une possibilité d'interruption et des messages courts complétés, à la demande, par des explications plus détaillées.

Les messages d'erreur sont un autre type habituel de messages. Quand des erreurs sont détectées, un effort doit être fait afin d'en indiquer la localisation, d'en donner la cause et de fournir un moyen approprié de correction. En conséquence, tout message d'erreur devrait indiquer ce qu'est l'erreur et ce que l'utilisateur peut faire pour la corriger.

Un exemple de conception erronée est un logiciel qui, lorsqu'il ne peut interpréter une commande, donne automatiquement une liste de toutes les commandes possibles. Une façon plus appropriée est d'imprimer un message court montrant que la commande n'a pas été comprise et conseillant l'utilisateur sur la manière d'obtenir les commandes disponibles en fonction de la transaction en cours.

Une autre notion importante est que la documentation papier (ou on-line) doit fournir des informations supplémentaires, et ne pas être seulement une traduction de codes d'erreur abscons.

c - Contenu des écrans

Si des informations doivent être affichées d'un écran à un autre, il ne faut pas afficher plus de 4 à 6 items. Les items longs constitués de caractères alphanumériques arbitraires doivent être affichés en groupes de 3 ou 4 séparés d'un espace (exception: les mots).

On peut utiliser des tirets (hyphens), mais pas de barres (slashes) car elles peuvent être confondues avec des caractères alphanumériques.

Les groupements doivent suivre les conventions d'usage habituelles.

Quand les informations affichées sont d'un intérêt potentiel de longue durée, il devrait y avoir un moyen facile de demander une impression, avec les contraintes de sécurité habituelles.

L'utilisateur ne devrait pas avoir à dépendre de sa mémoire ou à prendre des notes.

Le contenu d'un écran ne devrait pas changer à la suite d'une requête d'impression.

Une organisation consistante des diverses caractéristiques d'affichage doit être adoptée sous forme de standard, autant que possible pour tous les écrans (e.g., un endroit pour les titres d'écran, un autre pour les sorties, un autre pour les options de commande, les instructions, les messages d'erreur, etc.).

Quand il y a trop de données, celles-ci doivent être automatiquement divisées en pages d'écran séparées, chaque page avec un label approprié afin d'identifier ses rapports avec les autres pages. De plus, un moyen aisé de commande doit être fourni pour passer facilement d'une page à l'autre.

Des moyens doivent être adoptés afin de rendre les formats d'écrans aisément perceptibles par l'utilisateur (e.g., différents champs d'affichage ou fenêtres, utilisation d'espaces, etc.) afin de bien distinguer entre les catégories d'affichage.

Idéalement, chaque affichage doit fournir à l'utilisateur toute l'information nécessaire au point particulier de la transaction (par exemple, le titre doit être régénéré quand l'utilisateur passe d'un écran à un autre).

Chaque affichage doit présenter seulement l'information essentielle pour la tâche, sans informations inutiles. A mesure que le nombre d'informations sur écran augmente, le temps nécessaire à la prise d'information augmente aussi. Il est déconseillé d'utiliser plus de 90% des caractères disponibles sur un écran.

Pour des dialogues simples, chaque ligne d'affichage doit présenter un seul item à la fois (sauf pour remplissage de formes).

L'utilisateur doit pouvoir retirer les items qui ne lui sont pas utiles.

d - Labeling

Le labeling consiste à placer un titre descriptif, une phrase, ou un mot adjacent a un groupe d'items reliés entre eux.

De bons labels permettent une identification rapide et peuvent aider l'utilisateur à parcourir plus vite l'écran, à la recherche d'items qui l'intéressent, ou pour aider à s'assurer qu'un item est entré dans le champ adéquat.

Des labels peuvent être utilisés pour identifier un seul item, un groupe d'items, ou tout un écran. Par exemple:

- quand l'utilisateur participe à la sélection des données à afficher, chaque écran doit avoir un label d'identification unique, un code alphanumérique ou une abréviation pour faciliter les demandes d'affichage de l'utilisateur,

- chaque menu devrait avoir un titre qui reflète la question pour laquelle une réponse est recherchée. S'il est difficile de déterminer un titre pour un menu, cela peut vouloir dire que le menu ne reflète pas un ensemble d'alternatives logiques et homogènes, et que cela nécessite une re-conception,

- quand des items sont affichés selon un format réparti, chaque champ devrait avoir un label associé pour l'identifier,

- il convient d'établir des labels pour chaque item. Il ne faut pas croire que l'utilisateur va toujours identifier un item particulier parce qu'il l'a rencontré dans le passé. Le contexte joue ici un rôle important. Par exemple, l'item 301-631-1948 pourra être reconnu comme numéro de téléphone s'il apparaît dans un annuaire, mais il peut ne pas être reconnu sur un écran non formaté.

Certaines règles de position et de typographie doivent être respectées. Par exemple:

- pour des champs à entrée unique, le label doit être à gauche du champ,

- pour des champs à entrées multiples, le label devrait être au dessus du champ,

- les labels et leurs champs <associés doivent être séparés d'au moins un espace,

- généralement, la ponctuation au sein des labels n'est pas recommandée,

- les unités de mesure doivent etre incorporées aux labels.

Nombre de recommandations présentées précédemment à propos du codage et de la dénomination, s'appliquent aussi au labeling, bien que, contrairement aux précautions observées pour la conception de codes, il ne soit pas nécessaire de former des labels à partir d'un vocabulaire réduit. Il vaut mieux développer des labels en utilisant tous les types de caractères (alphabétiques, numériques, ou symboliques) qui ont une signification pour l'utilisateur. Cependant, il est convenu de ne pas établir de labels trop longs (jusqu' à 8 caractères) afin de réduire la densité des écrans.

Bien évidemment, comme pour les commandes, les labels doivent être clairs et uniques: même de légères ressemblances entre labels augmentent la confusion.

Si un label est un code, alors ce code doit être signifiant pour l'utilisateur. Quand il y a un doute quant à la familiarité d'un code, il vaut mieux ne pas l'utiliser et en tester un autre.

Les labels doivent aussi suivre les règles de cohérence: les labels doivent être distincts les uns des autres. Une construction grammaticale constante doit être utilisée (il ne faut pas un mot à un endroit et une phrases courtes ailleurs).

Au sein d'un même écran, les labels doivent être mis en valeur, soulignés, distincts en position et en format pour aider à distinguer les informations affichées des autres types d'affichage (e.g., messages d'erreurs, prompts, etc.). Un formatage consistant et un alignement des labels permettent aussi de minimiser le temps de recherche.

D'un écran à un autre, l'ordre et la disposition spatiale doivent être consistants, y compris le format interne détaillé pour les champs fréquemment utilisés.

Pour des raisons de flexibilité, avec les écrans souvent utilisés, on peut laisser l'utilisateur définir lui-même ses identificateurs d'écran. Bien que positionner ces identificateurs dans le coin en haut, à gauche de l'écran, on peut aussi permettre à l'utilisateur de définir lui-même ce positionnement. Cette identification d'écran doit être assez courte (3 à 5 caractères) et suffisamment signifiante pour être facilement rappelée.

Pour les mêmes raisons de flexibilité, une option devrait permettre de souligner les labels, par exemple, en augmentant leur intensité lumineuse pour de nouveaux utilisateurs, et en diminuant l'intensité des labels pour des expérimentés.

e - Groupement d'items

Les techniques de groupement sont importantes pour aider à l'organisation des informations sur écran. Les quatre critères principaux de groupement d'informations sont les suivants: ordre des items, fréquence d'utilisation, fonctionnalité, et importance.

Le groupement séquentiel est basé sur le principe d'afficher les items dans l'ordre selon lequel ils sont transmis ou reçus. Par exemple, si une tâche requiert l'entrée de données dans un certain ordre, alors l'écran doit être conçu afin de permettre l'entrée de ces données dans le même ordre. Pour les sorties, ce principe signifie que les items affichés le seront dans leur ordre d'utilisation. La première information nécessaire doit être affichée en haut de

l'écran, la deuxième en dessous, etc. Ce principe s'applique non seulement aux séquences d'items, mais aussi aux groupes d'items. Un exemple serait un processus dans lequel l'utilisateur doit vérifier toutes les données historiques avant de passer à d'autres sections d'écran. Les données historiques doivent non seulement être groupées ensemble, mais affichées de façon prégnante en haut de l'écran. L'ordre naturel des données est aussi une autre base pour le groupement séquentiel. Toutes choses égales par ailleurs, certaines séquences semblent plus naturelles que d'autres. Par exemple, les prénoms précèdent généralement les noms de famille, et les nombres sont habituellement placés en ordre croissant. La mise en oeuvre de regroupements séquentiels requiert évidemment une bonne connaissance des procédures de la tâche et une bonne formalisation des étapes de la tâche.

Le groupement selon la fréquence est basé sur le principe que les items qui sont le plus fréquemment utilisés doivent être regroupés. Si un écran était conçu uniquement selon ce critère, cela conduirait en fait à un ordre des items selon leur fréquence décroissante. Les items les plus utilisés seraient en haut de l'écran et les moins utilisés en bas. Une fois que les items sont regroupés de façon séquentielle, on peut ainsi placer les items utilisés le plus fréquemment en tête de chaque groupe. Dans le cas de groupements par fréquence, il est nécessaire de disposer d'une simulation ou d'une description précise des opérations de la tâche afin d'obtenir la fréquence relative des items.

Le groupement fonctionnel est une autre possibilité, qui consiste à grouper tous les items qui correspondent à un type particulier d'activité.

Par ailleurs, le groupement par importance peut être utilisé. Si certains items sont critiques, ils doivent être placés en premier ou à un endroit où ils ne pourront être négligés.

f - Standardisation

Ceci est un point très important qui peut être satisfait assez facilement, si une attention particulière est apportée à la conception. La règle est que, sauf

si cela contredit d'autres exigences ou crières (e.g., groupement), il convient de s'assurer que les items sont standardisés sur un même écran, mais aussi entre plusieurs écrans. Les items qui contiennent la même information doivent être dénommés et présentés de façon identique, sous le même format. D'un autre côté, un effort doit être fait pour ne pas dénommer de la même façon des items qui ne sont pas strictement identiques.

De façon générale, le codage, la dénomination, les titres, les groupements, la longueur des champs, etc., doivent être standardisés. Dans cet objectif, il est utile de constituer une grille d'analyse des divers éléments de l'interface selon les règles mentionnées afin de s'assurer de leur cohérence.

g - Texte

Le texte doit être affiché en capitales et minuscules. L'utilisation de capitales est recommandée quand il est nécessaire d'attirer l'attention de l'utilisateur ou à cause d'une mauvaise lisibilité des minuscules.

Le texte devrait être justifié à gauche, en laissant les marges de droite non justifiées. Excepté quand la justification à droite peut se faire par espacement variable, il faut maintenir un espacement proportionnel constant au sein des mots, et un espacement constant entre les mots.

Les mots doivent être affichés intégralement, avec une coupure de mots minimale.

Les paragraphes doivent être séparés par au moins une ligne blanche. Toutes les recommandations habituelles concernant la clarté des textes écrits s'appliquent ici aussi.

h - Listes

Quand de longues listes d'items doivent être affichées, elles devraient suivre un ordre logique. S'il y a des colonnes multiples, leur ordre doit être vertical pour chaque colonne.

Si des informations doivent être mémorisées, il faut mettre les plus difficiles à mémoriser au début, les plus faciles au milieu et celles qui doivent être rappelées pour la transaction immédiatement suivante à la fin.

Les colonnes de données numériques doivent être justifiées à droite ou justifiées en fonction d'un point décimal fixe.

Excepté pour les listes courtes de 4 à 5 items, afin de compacter le format d'affichage, et si cela est fait de manière consistante, les listes alphabétiques doivent être alignées verticalement et justifiées à gauche afin de permettre un balayage visuel rapide.

L'indentation peut être utilisée pour souligner des éléments subordonnés dans les listes hiérarchiques.

Les listes de données doivent être organisées dans un ordre reconnaissable afin de faciliter le balayage visuel et l'assimilation.

Quand des données listées sont identifiées par des nombres ou des lettres, le format de telles listes doit être distinct des listes d'options de menus. Avec des nombres, les labels doivent commencer par 1, pas par 0. Pour la numérotation hiérarchique, des nombres complets doivent être utilisés, sans omettre les éléments redondants.

i - Tables

Dans les affichages de tables, les lignes et les colonnes doivent être libellées selon les mêmes règles que pour les champs ou formulaires d'entrée de données.

Dans les tables denses comportant beaucoup de lignes, une ligne blanche (ou autre caractéristique visuelle distinctive) doit être insérée toutes les 5 lignes comme aide au balayage visuel horizontal.

Quand il y a plus d'une colonne, les colonnes doivent être séparées par au moins 3 ou 4 espaces si elles sont justifiées à droite, et au moins 5 espaces autrement. L'espacement des colonnes doit être consistant d'un écran à l'autre.

Quand des tables sont utilisées pour des objectifs de référencement, comme pour un index, les éléments indexés doivent être affichés dans la colonne de gauche. Les éléments les plus pertinents pour la réponse de l'utilisateur dans la colonne adjacente suivante, et les éléments associés, mais moins pertinents, plus loin à droite.

j - Défilement/ fenêtres

Quand une fenêtre doit être utilisée pour le balayage visuel de données, la fenêtre doit être plus grande qu'une seule ligne.

Les formats d'affichage doivent être changés seulement pour distinguer entre des sous-tâches ou des types d'activité.

Le corps de l'écran, quand il est utilisé pour l'affichage de données, doit être formaté afin de présenter les informations de façon cohérente, sans de nombreuses partitions de type fenêtre.

Chaque écran doit contenir un titre décrivant brièvement son contenu. Ce titre doit être séparé d'une ligne du corps de l'écran.

Les dernières lignes de l'écran doivent être réservées pour les messages de statut et d'erreur, les prompts, et pour l'entrée de commandes.

Quand les données dépassent la capacité d'un seul écran, l'utilisateur devrait avoir un moyen facile de se déplacer en avant et en arrière parmi les divers écrans, par défilement de type "paging" ou "scrolling".

Quand il y a plusieurs pages d'écran, chaque page d'écran doit comporter (e.g., en haut à droite) un identificateur numérique (sans zéro de début) ainsi que le nombre total de pages d'écran.

Le "paging/scrolling" est acceptable quand l'utilisateur recherche des éléments spécifiques, et pas quand il doit discerner une relation parmi divers ensembles de données.

Quand une liste d'éléments numérotés excède une page d'écran et qu'il faut utiliser du paging/scrolling pour continuer, les items doivent être numérotés de façon continue en relation avec le premier item du premier affichage.

Pour les écrans de tables, les têtes de colonnes et de lignes doivent être préservées dans chaque portion de l'écran affiché.

Quand des listes ou des tables de longueur variable excèdent la capacité d'un seul écran, leur continuation ou leur fin doit être explicitement indiquée sur l'écran (e.g., continue sur page suivante, fin de liste), excepté pour les listes courtes dont la fin est évidente.

Une orientation consistante doit être utilisée selon que les données sont représentées comme se déplaçant derrière une fenêtre d'affichage fixe (scrolling) ou bien que la fenêtre d'affichage est représentée comme se déplaçant au-dessus d'un ensemble fixé de données (windowing).

L'utilisateur peut s'adapter à l'un ou l'autre de ces concepts, s'il est maintenu de façon consistante. Cependant, le windowing semble plus naturel pour des sujets inexpérimentés, causant moins d'erreurs.

Dans les applications ou le curseur est déplaçé de façon libre au sein d'une page de données affichées, le windowing doit être sélectionné plutôt que le scrolling comme base conceptuelle du mouvement de l'écran.

Quand une orientation de windowing ou scrolling est maintenue de façon consistante, la dénomination des fonctions de scrolling doit se référer à la page affichée (ou à la fenêtre) et pas aux données affichées (e.g., la fonction "suivant", paramétrée à 10 devrait signifier que 10 lignes de l'écran vont disparaître du bas de l'écran et que 10 lignes précédentes vont apparaître en haut de l'écran).

Quand l'utilisateur peut être exposé à différents systèmes adoptant des concepts différents, toute référence à des fonctions de scrolling doit éviter des dénominations qui impliquent une orientation spatiale (e.g., dessus et dessous), mais plutôt utiliser de façon consistante des termes tels que avant et arrière (ou bien précédent et suivant) pour se référer aux mouvements au sein d'un ensemble de données affichées. Si ce sont des touches fonctions, elles doivent comporter des flèches plutôt que des termes.

k - Rafraîchissement d'écrans.

Si cela est compatible avec les exigences opérationnelles, l'utilisateur doit être capable de demander un rafraîchissement automatique (computer regeneration) des données affichées, et de contrôler sa vitesse.

Les changements de valeurs de données que l'utilisateur doit lire de façon précise doivent être affichés seulement dans une position fixe sur l'écran, et pas plus vite qu'une par seconde.

Quand l'écran est automatiquement rafraîchi, l'utilisateur doit pouvoir arrêter ce processus (stopper, figer) à tout moment, et examiner les données de façon plus délibérée. Dans certaines applications, il peut être utile de laisser l'utilisateur passer d'un écran à un autre, en avant ou en arrière pour examiner les données.

Quand un écran est en cours de rafraîchissement, un label approprié doit être ajouté afin de rappeler ce statut à l'utilisateur.

Quand un écran est rafraîchi en temps réel, l'utilisateur doit être averti si des changements significatifs (mais non affichés) sont détectés dans le

traitement par le logiciel des données nouvelles. Par ailleurs, le retour de l'écran doit se faire en temps réel, à moins que l'utilisateur ne spécifie autre chose.

Quand des écrans standard de données sont utilisés pour des objectifs particuliers, l'utilisateur doit pouvoir supprimer temporairement l'affichage des données non nécessaires pour l'étape de la tâche dans laquelle il se trouve.

Quand c'est le cas, l'utilisateur doit pouvoir récupérer rapidement l'écran dans sa configuration d'origine. Dans certaines applications, cela peut être fait automatiquement, après expiration d'un délai déterminé, plutôt que de dépendre de la mémoire de l'utilisateur.

VI - BIBLIOGRAPHIE

Quelques références générales

Badre, A., and Shneiderman, B. (1982) Directions in human/computer interaction. Norwood, NJ: Ablex.

Bailey, R. W. (1982) Human performance engineering: a guide for system designers. Englewood Cliffs, NJ: Prentice-Hall.

Barmack, J. E., and Sinaiko, H. W. (1966) Human factors problems in computer-generated graphic displays (Study S-234), Washington DC: Institute for Defense Analyses (AD-63617C).

Brown, C. M., Burkleo, H. V., Mangelsdorf, J. E., Olsen, R. A., and Williams, A. R., Jr. (1981) Human factors engineering standards for information processing systems. Sunnyvale, CA: Lookeed Missiles and Space Company.

Card, S. K., Moran, T. P., and Newell, A. (1983) The psychology of human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum.

Engel, S. E., and Granda, R. E. (1981) Guidelines for man-display interfaces. Technical Report TR 00.2720. Poughkeepsie, NY: IBM.

Foley, J. D., Wallace, V. L., and Chan, P. (1981) The human factors of graphics interaction: tasks and techniques. (Report No. GWU-IIST-81-3). Washington DC: George Washington University.

Gallitz, W. O. (1981) Human factors in office automation, Atlanta, GA: Life Office Management Association.

Gallitz, V. O. (1981) Handbook of screen format design. Wellesley, MA: Q.E.D. Information Science.

Hiltz, S. R., and Turoff, M. (1978) The network nation. Reading, MA: Addison-Wesley.

Martin, J. (1973) Design of man-computer dialogues. Englewood Cliffs, NJ: Prentice-Hall.

Miller, L. A. , and Thomas, J. C. Jr. (1976) Behavioral issues in the use of interactive systems (Report No. RC 6326). Yorktown Heights, NY: IBM.

Newman, W. M., and Sproull, R. F. (1979) Principles of interactive computer graphics. NY: Mc. Graw-Hill.

Parrish, R. N., Gates, J. L., Munger, S. F., and Sidorsky, R. C. (1982) Development of design guidelines and criteria for user/operator transactions with battlefield automated systems. Phase II: Prototype design handbook for combat and material developer. (Report WF-80-AE-00). Alexandria, VA: US Army Research Institute.

Pew, R. W., and Rollins, A. M. (1975) Dialog specification procédures. Report 3129. Cambridge, MA: Bolt Beranek and Newman.

Ramsey, H. R., and Atwood, M. E. (1979) Human factors in computer systems: a review of the literature. (Technical Report No. SAI-79-111-DEN). Englewood CO: Science Applications.

Scapin, D. L. (1979) Ergonomie des dialogues homme-ordinateur. Rapport IRIA CA 79-09-RQ01, Le Chesnay.

Scapin, D. L. (Ed.) (1983) Ergonomie de l'informatique. Bulletin de Liaison de la Recherche en Informatique et Automatique, No. 85. Le Chesnay: I.N.R.I.A.

Seibel, R. (1972) Data entry devices and procedures. In Van Cott, H., and Kincade, R. G. (Eds.). Human engineering guide to equipment design. Washington DC: US Government Printing Office, 311-344.

Shackel, B. (1981) Man-computer interaction: human factors aspects of computers and people. Sijthoff and Hoordoff.

Shneiderman, B. (1980) Software psychology: human factors in computer and Information systems. Cambridge, MA: Winthrop.

Smith, S. L., and Aucella, A. F. (1983) Design guidelines for user-interface to computer-based information systems. Report No. ESD-TR-83-122. Bedford, MA: MITRE.

Spérandio, J. C., et Scapin, D. L. (1985) Aspects ergonomiques de la communication homme-machine. In Principes de la communication homme-machine: parole, vision et langage naturel. Le Chesnay: Cours et Séminaires I.N.R.I.A.

Thomas, J. C., and Schneider, M. L. (1984) Human factors in computer systems. Norwood, NJ: Ablex.

Vassiliou, Y. (1982) Human factors and interactive computer systems. Norwood, NJ: Ablex.

Williges, B. H., and Williges, R. C. (1984) Dialog design considerations for interactive computer systems. In Muckler, F. A. (Ed.). Human Factors Review. Santa Monica, CA: The Human Factors Society.

Quelques revues scientifiques

- Behaviour and Information Technology
- Communications of the ACM.
- Ergonomics.
- Human factors.
- International Journal of Man-machine Studies.

Quelques numeros spéciaux de revues

- ACM Computing Surveys, vol. 13, no. 1, July 1981. The psychology of the computer user.

- IBM System Journal, vol. 20, No. 2, 1981. The human side of computers.

- International Journal of Man-Machine Studies, vol. 15, no. 1, July 1981. The semantics and syntax of human-computer interaction.

Quelques conférences

- Human Factors in Computer Systems, Gaithersburg, MD, USA (1982)

- Human Factors in Computing Systems, Boston, MA, USA (1983), San Francisco, CA, USA (1985)

- INTERACT (IFIP Conference on Human-Computer Interaction), London, UK (1984)

- Human Factors Society Annual Meetings (USA) (les 5 dernières années)

VII - INDEX

Mots-clefs.....Sections

VIII - AIDE-MEMOIRE

Cette liste de questions n'est qu'indicative.

Pré-requis

Les caractéristiques des utilisateurs potentiels ont-elles été identifiées?

Quel est le niveau d'expérience des utilisateurs?

Existe-t-il un mécanisme permettant aux utilisateurs d'effectuer des remarques, des suggestions?

Dispose-t-on d'une analyse précise des tâches?
En particulier, les caractéristiques des informations et objets manipulés dans ces tâches ainsi que les opérations sur ces informations et objets sont-elles connues?

A-t-on examiné la question du partage des tâches entre fonctions informatiques et opérations de l'utilisateur?

Le séquencement entre événements, présentations d'informations, actions de l'opérateur est-il déterminé?

S'est-on donné les moyens d'évaluer et de tester le logiciel aux étapes importantes de la conception?

Critères de conception

La compatibilité entre écrans et documents papier, entre les informations présentées et les attentes des utilisateurs, entre les dénominations et le vocabulaire de l'utilisateur, etc., est-elle respectée?

L'homogénéité, la cohérence entre les diverses présentations d'informations, entre les diverses commandes, entre les procédures, etc. sont-elles respectées?

S'est-on assuré de la concision des informations présentées, des commandes disponibles?

Le logiciel dispose-t-il de plusieurs niveaux correspondant à l'expérience des utilisateurs?

Chacune des entrées prévisibles de l'utilisateur comporte-t-elle un feed-back approprié?

L'utilisateur, notamment naïf, dispose-t-il d'un guidage ?

S'est-on assuré que les actions de l'utilisateur sont minimisées, en nombre et en durée?

Chacune des actions possibles de l'utilisateur ne peut-elle se faire que de façon explicite?

S'est-on assuré d'une bonne gestion des erreurs?

LE DIALOGUE

Types de dialogues

Avant de choisir un mode de dialogue, a-t-on considéré l'expérience des utilisateurs, les caractéristiques des entrées (e.g., leur nombre, arbitraires ou non), le temps de réponse prévisible, etc.?

Menus

Combien de sélections simultanées peut-on effectuer sur les divers menus?

La sélection se fait-elle par pointage direct?

Comment sont constitués les codes des options?

Sur quoi repose l'ordre des options?

S'il y a une numérotation des options, dans quel ordre se fait-elle?

Le texte des options est-il justifié à gauche?

Les options fournies sont-elles pertinentes pour la transaction en cours?

Les labels des options sont-ils informatifs?

Chaque menu comporte-t-il un titre?

Le menu est-il sur plusieurs pages, selon quelle organisation?

Combien y-a-t-il d'étapes dans les menus?

Le feed-back du choix d'options est-il inférieur à 2 secondes?

Touches fonctions

Les touches fonctions sont-elles organisées spatialement?

Combien de fonctions correspondent à chaque touche fonction?

Quand il y a plusieurs fonctions assignées aux touches, le mode fonction active est-il affiché?

Utilise-t-on la double frappe pour des touches fonctions?

Les touches fonctions sont-elles codées de façon informative?

Les touches fonctions non pertinentes pour la transaction en cours sont-elles désactivées?

Les touches dont la fonction peut avoir des conséquences destructrices sont-elles protégées?

L'exécution des touches est-elle toujours signalée, avec un temps de réponse inférieur à 2 secondes?

Langage de commande

L'aire d'entrée de commandes est-elle bien distincte?

Le nombre de commandes est-il aussi réduit que possible?

Chaque commande est-elle unique et distincte des autres?

Le langage de commande est-il structuré?

Le langage de commande comporte-il un guidage?

Une ponctuation est-elle nécessaire?

Le nombre d'espaces est-il une contrainte?

Le langage comporte-t-il des quantificateurs?

Le paramétrage se fait-il par mots-clefs ou est-il positionnel?

Que se passe-t-il quand des paramètres manquent?

Y-a-t-il des valeurs par défaut?

Ces valeurs par défaut sont-elles affichées?

Les fonctions commandes sont-elles dénommées à partir de termes communément admis?

Le langage de commande est-il logique, précis, homogène?

Le langage de commande est-il flexible, permissif?

Procédures de dialogue

Les séquences de commande correspondent-elles à l'ordre des opérations requises par la tâche?

La longueur et la complexité des procédures de dialogue permettent-elles de limiter la charge en mémoire?

L'utilisateur peut-il définir lui-même des procédures de dialogue?

Le logiciel comporte-t-il une introduction aux procédures?

L'information sur le statut de l'interaction est-elle affichée?

Est-il possible d'accéder à l'historique du dialogue?

Quand il y a plusieurs niveaux d'affichage, leur nombre est-il minimisé?

La position au sein de la transaction en cours est-elle affichée?

ENTREES

L'entrée d'informations se fait-elle par sur-frappe?

L'entrée comporte-t-elle un guidage concernant la longueur maximale des items et leur caractère optionnel?

Les passages d'un mode d'entrée à un autre sont-ils minimisés?

L'utilisateur doit-il entrer des informations déjà entrées?

Les zéros présents dans les entrées sont-ils optionnels?

L'utilisateur doit-il faire la distinction entre plusieurs espaces?

Existe-t-il des valeurs par défaut?

Comment s'effectue leur confirmation?

Les entrées ne sont-elles pas trop longues?

Utilise-t-on des abréviations?

Les abréviations sont-elles conformes à l'usage?

Quelle méthode de construction d'abréviations est utilisée?

Les utilisateurs peuvent-ils définir eux-mêmes leurs abréviations?

Y-a-t-il plus d'une abréviation par item ou commande?

Sur quelle base a été défini l'ordre des informations à entrer?

Lors de l'édition, les données anciennes ainsi que les nouvelles sont-elles affichées?

L'achèvement de l'entrée d'informations se fait-elle de façon explicite?

Toutes les données entrées sont-elles affichées?

L'entrée des données est-elle guidée par des labels de champs et des messages d'aide?

Le logiciel informe-t-il l'utilisateur de ses erreurs en en donnant la cause, l'endroit et une façon d'y remédier?

Quelle est la forme du curseur?

Y-a-t-il un clignotement? Quelle est sa fréquence?

Le curseur est-il facile à localiser?

Y-a-t-il des interférences entre le curseur et les caractères qu'il dénote?

Le curseur est-il toujours placé au premier caractère du premier champ de chaque écran?

Le curseur est-il automatiquement déplacé au champ suivant après la fin d'une entrée sur le champ précédent?

Séquences de commande

Les séquences de commande sont-elles simplifiées au maximum?

Les séquences de commande correspondent-elles à l'expérience des utilisateurs?

Les expérimentés peuvent-ils définir eux-mêmes des macro-commandes?

L'achèvement des séquences de commande se fait-il toujours de façon explicite?

Quand des entrées sont différées, le clavier est-il bloqué et l'utilisateuur prévenu?

La forme et les conséquences des séquences de commande sont-elles maintenues de façon homogène tout au long des transactions?

Si les effets d'une suite de commandes diffèrent en fonction du contexte, ce contexte est-il affiché?

L'utilisateur peut-il interrompre, différer, annuler une transaction de façon consistante?

Existe-t-il une fonction d'annulation?

Existe-t-il une fonction de retour-arrière?

Existe-t-il une fonction de restart?

Existe-t-il une fonction de destruction?

Existe-t-il une fonction de fin?

La prise en compte des commandes est-elle confirmée par leur exécution ou par un message?

Existe-t-il une fonction affichant les entrées précédentes?

Le feed-back est-il homogène tout au cours du dialogue?

Quand une erreur est détectée, quelles informations sont affichées?

Quand une erreur est corrigée par l'utilisateur, doit-il à nouveau effectuer une commande d'entrée explicite?

Quand un paramètre est erroné, l'utilisateur doit-il entrer à nouveau toute la séquence?

Quand une valeur par défaut est incluse dans une séquence, est-elle affichée?

Quand une commande est potentiellement destructrice, doit-elle être confirmée?

La commande ou touche de confirmation est-elle distincte de la touche de validation?

SORTIES

Le temps de réponse dépasse-t-il 15 secondes?

Quelle est la variabilité du temps de réponse (supérieur à 5%)?

Quand le temps de réponse est long, un message indiquant ce qu'effectue le logiciel est-il affiché?

Dispose-t-on des bonnes informations à afficher?

A-t-on utilisé une méthode pour s'en assurer?

Les informations sur écran sont-elles visibles, lisibles, consistantes, concises, immédiatement utilisables?

Les messages sont-ils adaptés au niveau des utilisateurs?

L'utilisateur peut-il changer de niveau de message?

Dispose-t-il d'une fonction d'aide?

A-t-on apporté une attention particulière aux messages d'erreur?

Faut-il consulter la documentation pour comprendre les messages d'erreur?

Dans quel but les codages sont-ils utilisés?

Chaque code est-il unique, familier?

Utilise-t-on une règle systématique d'encodage?

La taille des codes est-elle minimisée?

Les codes sont-ils composés de combinaisons inhabituelles de lettres?

Les codes sont-ils prononçables?

Les codes sont-ils tous optimisés? Si non, en fonction de quelle priorité?

Les codes, quand il s'agit d'affichages, sont-ils présentés de façon consistante?

Le contexte d'utilisation des codes est-il identique à celui utilisé lors de l'apprentissage?

Des codes mixtes sont-ils utilisés?

Les codes affichés sont-ils utilisés immédiatement?

Les codes ont-ils été constitués à partir d'opinions d'utilisateurs?

Un codage auxiliaire est-il adopté pour les items importants?

Le codage par localisation est-il utilisé pour distinguer les informations?

Les caractères alphabétiques et numériques présents dans un code sont-ils regroupés?

Les codes sont-ils signifiants ou arbitraires?

Les codes sont-ils consistants d'un écran à un autre, d'une transaction à une autre?

Les symboles sont-ils analogues aux éléments qu'ils représentent?

Sont-ils familiers?

Combien de couleurs sont utilisées?

Dans quels cas le codage couleur est-il employé?

Les affichages ont-ils été formatés de façon efficace avant d'y ajouter un codage couleur?

Le codage couleur est-il redondant avec d'autres codes?

Combien de valeurs le codage intensité comporte-t-il?

A quoi est utilisée l'inverse vidéo?

Le clignotement fonctionne-t-il sur l'item désigné ou sur un soulignement?

Est-il utilisé seulement pour des événements critiques?

Les durées de clignotement/non clignotement sont-elles égales?

Y-a-t-il un moyen d'arrêter le clignotement?

S'arrête-t-il suite à une action de l'utilisateur?

Les instructions destinées à l'utilisateur précèdent-elles toujours les listes de choix proposés?

Chaque label est-il unique?

Les labels correspondent-ils bien au contenu qu'ils dénotent?

Chaque groupe d'information, message ou écran possède-t-il un titre, un descriptif de son contenu?

Les labels sont-ils situés de façon adjacente aux éléments qu'ils décrivent?

Les labels sont-ils soulignés, mis en valeur?

La méthode de codage des labels est-elle distincte des autres catégories d'affichage?

Les unités de mesure sont-elles incorporées aux labels?

Les labels comportent-ils une ponctuation?

Les labels sont-ils utilisés de façon homogène en contenu et en format tout au long du dialogue?

Sur l'écran, les informations affichées sont-elles toutes nécessaires?

Seules les options, commandes ou fonctions disponibles sont-elles affichées?

La densité d'information est-elle réduite au minimum, en particulier pour les opérations critiques?

Les informations sont-elles présentées selon un format directement utilisable?

Les documents papier sont-ils conçus de sorte qu'il y a une similitude suffisante avec les formats d'écran?

Les formats d'affichage sont-ils consistants?

Les formats d'écran permettent-ils un accès facile aux informations?

Si des informations doivent être affichées d'un écran à un autre, y-en-a-t-il plus de 4 à 6?

Les items longs sont-ils divisés en items plus courts?

Les slashes sont-ils utilisés dans les affichages?

Y-a-t-il au minimum un espace au-dessus et en-dessous des informations critiques?

Y-a-t-il au moins deux espaces de chaque côté?

Les aires d'affichage sont-elles spécialisées pour les différentes catégories d'items?

Les utilisateurs peuvent-ils supprimer de l'écran les items non pertinents?

Les champs sont-ils définis de façon à utiliser le même moyen d'entrée avant de passer à un autre?

Quand les informations ont un intérêt de longue durée, peut-on les imprimer facilement?

Suite à une impression, l'écran subit-il des modifications?

Les items sont-ils regroupés séquentiellement, selon leur fréquence d'utilisation, leur fonctionnalité, leur importance?

Ces groupements sont-ils bien distincts?

Les items, formats, champs et écrans sont-ils standardisés?

Les informations identiques sont-elles affichées toujours au même endroit?

Le texte est-il affiché en minuscules et majuscules?

Est-il justifié à gauche?

La coupure de mots est-elle minimale?

Les listes suivent-elles un ordre logique?

Les colonnes de données numériques sont-elles justifiées à droite ou en fonction de la virgule ou du point décimal?

L'indentation est-elle utilisée pour indiquer des éléments subordonnés?

Chaque item est-il sur une nouvelle ligne quand il y a énumération?

Les tables denses comportent-elles des séparateurs de ligne?

Les colonnes sont-elles séparées d'au moins 4 à 5 espaces?

Les index sont-ils affichés sur la colonne de gauche?

Les items à comparer caractère par caractère sont-ils situés les uns au-dessus des autres?

De quelle taille sont les fenêtres?

Le découpage en fenêtres est-il limité à un minimum?

Les utilisateurs peuvent-ils afficher la page entière sur laquelle ils travaillent?

Quand il y a plus d'une page, les pages suivantes sont-elles numérotées en fonction de la première page?

Y-a-t-il un message indiquant qu'il y a plusieurs pages?

Existe-t-il une fonction de rafraîchissement d'écran?

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique