

logiciels interactifs et ergonomie

modèles et méthodes de
conception

par

Marie-France BARTHET
Professeur à l'Université
des Sciences Sociales
de Toulouse

1988

Version électronique
Lire l'avertissement

Résumé

(extrait du quatrième de couverture)

L'objectif de ce livre est de montrer qu'il est possible de réaliser des logiciels interactifs beaucoup plus adaptés aux utilisateurs en modifiant les méthodes d'analyse et de programmation de manière à intégrer des éléments provenant de l'ergonomie ou de la psychologie cognitive.

L'auteur se situe dans le cadre de l'informatique des organisations et des applications collectives de type professionnel ou grand public qui connaissent actuellement un grand essor.

Cet ouvrage s'adresse à tous ceux qui interviennent dans le processus de conception des applications interactives, que ce soient les informaticiens-analystes, les conseillers informatiques, les ergonomes ou les formateurs.

Avertissement

L'ouvrage de Marie-France BARTHET "Logiciels interactifs et ergonomie, modèles et méthode de conception" est paru en 1988 aux éditions Dunod Informatique (Bordas, Paris) avec l'ISBN 2-04-018726-X. L'ouvrage original comporte 219 pages, une préface, une bibliographie, et un glossaire. Il est actuellement épuisé chez cet éditeur, il reste cependant encore quelques exemplaires disponibles chez certains libraires.

Cette version électronique de l'ouvrage a été réalisée en 1999 par Francis JAMBON, à partir des fichiers informatiques originaux, avec l'accord et également l'aide précieuse de son auteur. L'ouvrage vous est présenté en version limitée, en effet la préface et une partie du résumé n'ont pu être retrouvés. Il se peut également que certaines parties soient d'une version antérieure à l'ouvrage paru chez Dunod Informatique, ou que des erreurs se soient glissées dans le processus de conversion aux formats HTML et PDF. En outre, la pagination a subi quelques modifications. C'est pourquoi il est préférable, en cas de doute sur le contenu ou dans les citations, de se référer à l'ouvrage paru chez Dunod Informatique.

Vous pouvez consulter et/ou télécharger cet ouvrage pour votre usage personnel uniquement, "à des fins scientifiques et éducatives dans un esprit universitaire". Il ne doit pas en être fait de copies ni de diffusions sous quelque forme que ce soit sans l'accord explicite de son auteur, Marie-France BARTHET.

L'ouvrage, quelque soit sa forme, n'est pas libre de droits.

"Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de l'auteur, ou de ses ayant-droits, ou ayant-cause, est illicite (loi du 11 mars 1957, alinéa 1^{er} de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. La loi du 11 mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective d'une part, et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration"

Sommaire

Préambule.....	7
Chapitre 1 : L'INTERFACE HOMME-ORDINATEUR.....	8
1.1 La conception usuelle des logiciels interactifs.....	8
1.1.1 Les limites des méthodes actuelles.....	8
1.1.2 Les grands mythes.....	14
1.1.3 L'apport d'autres disciplines.....	16
1.1.4 Les différents points de vue.....	18
1.2 Les principaux concepts développés.....	21
1.2.1 Les concepts classiques.....	21
1.2.2 Les nouveaux concepts sur les applications interactives et les utilisateurs.....	23
1.3 Présentation de l'ouvrage.....	28
1.3.1 Composition générale.....	28
1.3.2 Comment le lire?.....	29
Chapitre 2 : LES STRUCTURES PROFONDES DES APPLICATIONS INTERACTIVES.....	31
2.1 La psychologie cognitive et l'utilisateur de logiciel.....	32
2.1.1 Eléments pertinents.....	32
2.1.1.1 L'image opérative.....	32
2.1.1.2 La planification hiérarchique.....	33
2.1.1.3 Utilisateurs expérimentés et débutants.....	35
2.1.1.4 Tâche et activité.....	37
2.1.2 Possibilités d'extrapolation.....	38
2.2 La Représentation Conceptuelle de l'application interactive.....	41
2.2.1 Introduction.....	41
2.2.2 Les paramètres variables de la Représentation Conceptuelle.....	43
2.2.2.1 Principes de base.....	43
2.2.2.2 Prise en compte de la variabilité dans la R.C.....	49
2.2.2.3 Description détaillée de la R.C.....	52
2.2.2.4 Formalisme graphique.....	59
2.2.3 Les paramètres constants de la Représentation Conceptuelle.....	63
2.2.3.1 Aides au travail.....	64
2.2.3.2 Aides à l'apprentissage.....	65
2.2.3.3 Les possibilités d'évolution.....	66
Chapitre 3 : LES ASPECTS DE SURFACE DES APPLICATIONS INTERACTIVES.....	69
3.1 L'apport de la psychologie cognitive à la Représentation Externe.....	70
3.1.1 Les structures de la mémoire humaine.....	70
3.1.2 Expérience et automatisme.....	73

5.4.1.2 Utilisateurs.....	153
5.4.2 Conception du nouveau système.....	158
5.4.2.1 Conception de la Représentation Conceptuelle.....	158
5.4.2.2 Conception de la Représentation Externe.....	170
Chapitre 6 : DIANE METHODE DE DIAGNOSTIC.....	174
6.1 Diagnostic d'un logiciel interactif.....	174
6.1.1 Démarche.....	174
6.1.2 Grille d'analyse.....	175
6.2 Cahier des charges d'un logiciel interactif.....	182
6.2.1 Spécifications générales.....	183
6.2.2 Spécifications particulières à l'application.....	184
6.3 Etude de cas " RTR ".....	185
6.3.1 Analyse des écrans.....	185
6.3.2 Enchaînements d'écrans.....	192
BIBLIOGRAPHIE.....	197
1 - Informatique.....	197
2 - Ergonomie.....	198
3 - Livres divers.....	200
GLOSSAIRE.....	202

Préambule

L'objectif de ce livre est de montrer qu'il est possible de réaliser des logiciels interactifs beaucoup plus adaptés aux utilisateurs et qu'il faut pour cela modifier les méthodes d'analyse et de programmation de manière à intégrer des éléments provenant de l'ergonomie ou de la psychologie cognitive.

Nos propos se situent dans le cadre de l'informatique des organisations et concernent plutôt des applications collectives de type professionnel ou grand public (c'est-à-dire que nous n'avons pas fait un travail spécifique sur les systèmes interactifs d'aide à la décision ou à la conception).

Les applications interactives de type professionnel ou grand public connaissent à l'heure actuelle un grand essor et concernent donc un nombre d'utilisateurs de plus en plus important.

Ce simple fait rend d'autant plus nécessaire d'intégrer à ces logiciels tous les éléments permettant un apprentissage et une manipulation plus aisés pour des utilisateurs qui n'ont ni le besoin ni l'utilité de rentrer dans la logique informatique.

Mais l'argument fondamental qui amène à se préoccuper de l'optimisation de l'interaction homme-machine est un prolongement logique de la volonté d'automatisation.

En effet, l'objectif essentiel de l'automatisation est d'augmenter la rentabilité de l'organisation en reportant une partie du travail des personnes vers les machines et en augmentant, si possible, la fiabilité de l'ensemble.

Dans le cadre de l'informatique des organisations où les personnes ne peuvent jamais complètement être remplacés par des automates (comme c'est le cas pour certaines chaînes de production), la diminution maximale du travail humain et la fiabilité maximale de l'ensemble sont liées à la qualité de l'interaction homme-machine.

La dernière raison que l'on peut avoir de se préoccuper de l'interaction homme-machine est d'optimiser le travail de l'informaticien quand on sait qu'en moyenne 50% du code sert à la gestion de la communication homme-machine.

Ce livre s'adresse à toutes les personnes qui interviennent dans le processus de conception des applications interactives que ce soit de manière globale comme les informaticiens-analystes ou de manière partielle comme les conseillers extérieurs, les correspondants informatiques, les ergonomes, les formateurs.

Chapitre 1 :

L'INTERFACE HOMME-ORDINATEUR

Ce premier chapitre a un double but car il présente d'une part l'état de l'art dans la conception des applications interactives, d'autre part les principaux concepts développés dans ce livre.

1.1 La conception usuelle des logiciels interactifs

Ce paragraphe présente d'une part l'ensemble des pratiques, des tendances et des présupposés qui servent de guides dans la conception des applications interactives, d'autre part les apports que l'on peut attendre de disciplines autres que l'informatique.

1.1.1 Les limites des méthodes actuelles

La conception des logiciels interactifs est faite, dans la très grande majorité des cas, par les informaticiens-analystes qui disposent, pour cela, de méthodes d'analyse et de conception des systèmes d'informations automatisés.

Ces méthodes les amènent à "déduire" l'interface entre l'utilisateur et l'ordinateur de la logique de fonctionnement des systèmes d'information.

Pour illustrer ce propos, nous faisons référence ici à deux méthodes bien connues en France, MERISE et AXIAL, qui sont assez caractéristiques de la pratique développée dans le domaine de la spécification des applications interactives.

La différence essentielle sur les applications interactives entre ces deux méthodes tient au degré de détail des spécifications:

- quand on a fini d'appliquer la méthode MERISE (Tar,83), on a des spécifications détaillées des données et des traitements associés, mais on ne dispose d'aucune spécification précise sur le dialogue homme-machine; la conséquence immédiate est que l'essentiel de l'interaction homme-machine est définie au niveau de la programmation à un moment où il est très difficile d'intégrer le

point de vue de l'utilisateur car on est alors entièrement absorbé par les contraintes matérielles et logicielles.

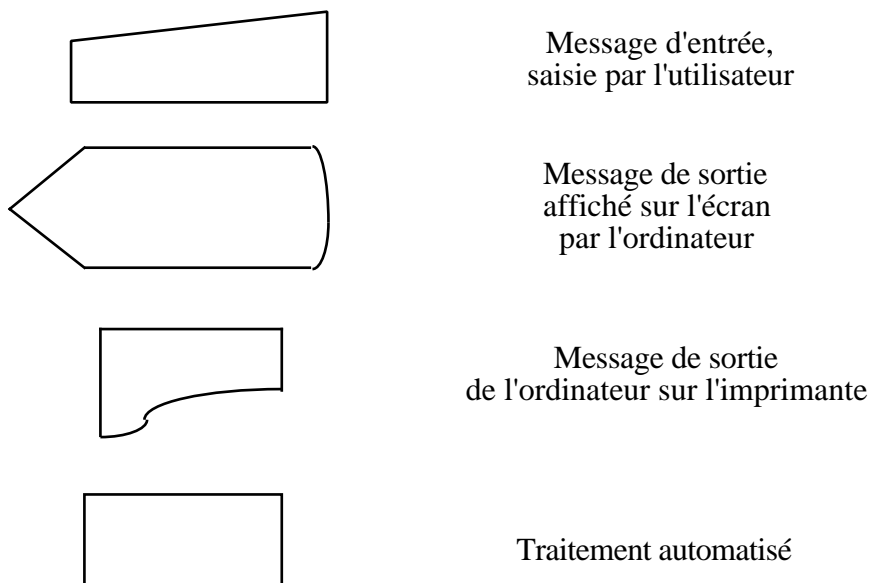
- par contre quand on applique la méthode AXIAL (Pe1,86) , on aboutit à un ensemble de spécifications précises concernant l'interaction homme-machine du point de vue de l'analyste sans prise en compte explicite de l'utilisateur.

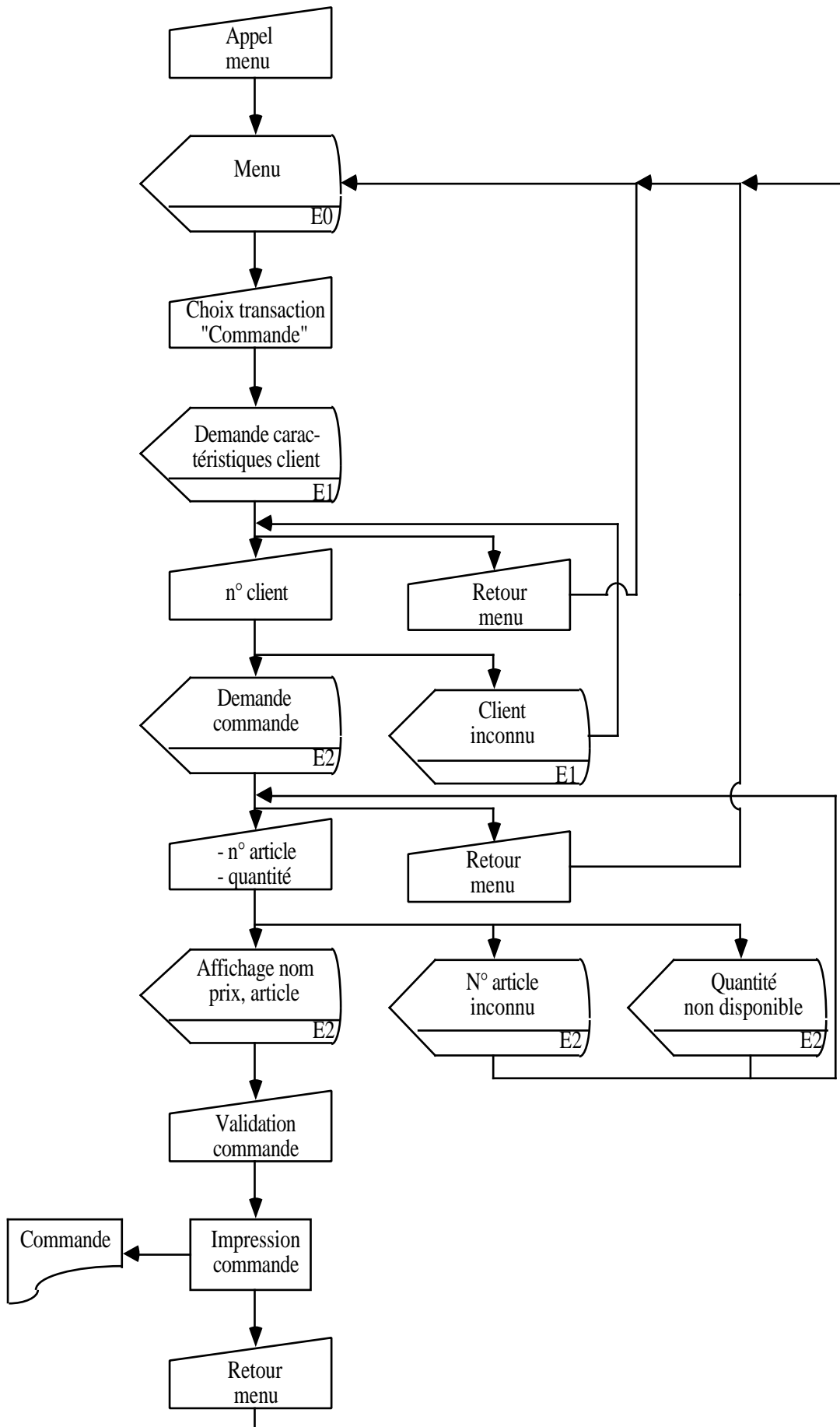
Pour illustrer l'absence de prise en compte du point de vue de l'utilisateur, nous prenons l'exemple suivant développé à partir du formalisme de la méthode AXIAL:

Cet exemple décrit un diagramme du dialogue utilisateur-machine qui visualise les enchaînements possibles entre un utilisateur et un programme "Commande" qui a pour rôle de permettre l'enregistrement des caractéristiques d'une commande.

Le cas normal, c'est-à-dire sans erreur, est présenté dans la partie verticale gauche; les erreurs ou les possibilités de sortir de cet enchaînement normal sont situées dans la partie droite de schéma.

La signification du formalisme est la suivante:





En observant ce diagramme, on peut se poser les questions suivantes:

- "Quels sont les critères utilisés pour concevoir le "menu" et sa hiérarchie, s'il y a lieu?"

Ces critères ne sont pas explicites mais si on regarde l'ensemble de la méthode, on voit que ce menu est déduit du découpage du système d'information (fonction puis opération puis poste de travail), mais qu'à aucun moment on ne se pose la question du point de vue de l'utilisateur c'est-à-dire "Quel est l'ensemble des opérations nécessaires à l'utilisateur pour qu'il puisse réaliser avec le plus de facilités possible sa tâche?".

Si on se pose ce type de questions, on peut être amené à regrouper des opérations appartenant à des fonctions différentes mais utilisées par une même personne, à créer une redondance à l'intérieur de la hiérarchie, à introduire des enchaînements propres à la logique d'utilisation de l'utilisateur... mais on a très peu de chance de retomber sur le découpage élaboré précédemment.

En d'autres termes, il est tout à fait exceptionnel que le point de vue de l'analyste et le point de vue de l'utilisateur coïncident.

En effet, le point de vue de l'analyste suit la *logique générale du système d'information* alors que le point de vue de l'utilisateur est guidé par la *logique d'utilisation de son poste de travail*.

- "Comment sont déterminés les enchaînements du dialogue?"

L'enchaînement proposé correspond-il à un enchaînement possible, probable, moyen?

En fait il est étonnant qu'il n'y ait qu'un type d'enchaînement possible (comme nous le verrons au chapitre 2) et ceci sans justification de ces enchaînements vis-à-vis d'une logique d'utilisation.

Y a-t-il possibilités d'interruption ou d'abandon en cours d'opération?

Ces possibilités systématiques ne sont pas prévues et au contraire, on voit que l'utilisateur est totalement contrôlé par le logiciel qui prévoit à quel moment précis il peut revenir au début du programme.

En d'autres termes le partage du pilotage de l'application entre l'homme et l'ordinateur n'est pas explicite, et, en conséquence, c'est le plus souvent le pilotage de l'ordinateur qui prime, empêchant ainsi l'utilisateur d'adapter ses enchaînements en fonction des contraintes propres à la réalisation de sa tâche.

- "En cas d'erreur quelles sont les possibilités de correction de l'utilisateur?"

On voit qu'il est prévu des enchaînements propres à la détection de certaines erreurs par le logiciel, mais on oublie qu'il y a des erreurs que l'ordinateur ne peut détecter alors que l'utilisateur peut les voir. Encore faut-il lui donner la possibilité de les rectifier en "défaisant" éventuellement une partie de ce qu'il a fait.

D'autre part, les erreurs détectées par le logiciel peuvent être d'origines différentes. Si c'est une erreur de saisie, les enchaînements prévus dans l'exemple sont bons, mais si c'est une erreur différente (numéro erroné sur le bordereau), la rectification de cette erreur va nécessiter de la part de l'utilisateur une stratégie différente (consultation d'une liste par exemple) qui est rendue impossible dans cet exemple.

- "Quelles sont les possibilités de guidage?"

A priori, le guidage n'est pas prévu dans ce dialogue utilisateur-machine.

Ce diagramme du dialogue utilisateur-machine est complété par les dessins d'écran.

Nous donnons ci-dessous, à titre d'exemple, un dessin d'écran:

Dessin de l'écran E2

The diagram shows a user interface screen with the following elements:

- Input fields for "Numéro client" (7 characters) and "Date" (8 characters).
- An input field for "Nom client" (20 characters).
- A table with 4 columns: "Numéro" (5 characters), "Nom article" (20 characters), "Quantité" (5 characters), and "P.U." (5 characters).
- Three rows of input fields for the table.
- A label "COM/03" in the bottom left corner.
- A "Validation" button with a checkmark icon in the bottom right corner.

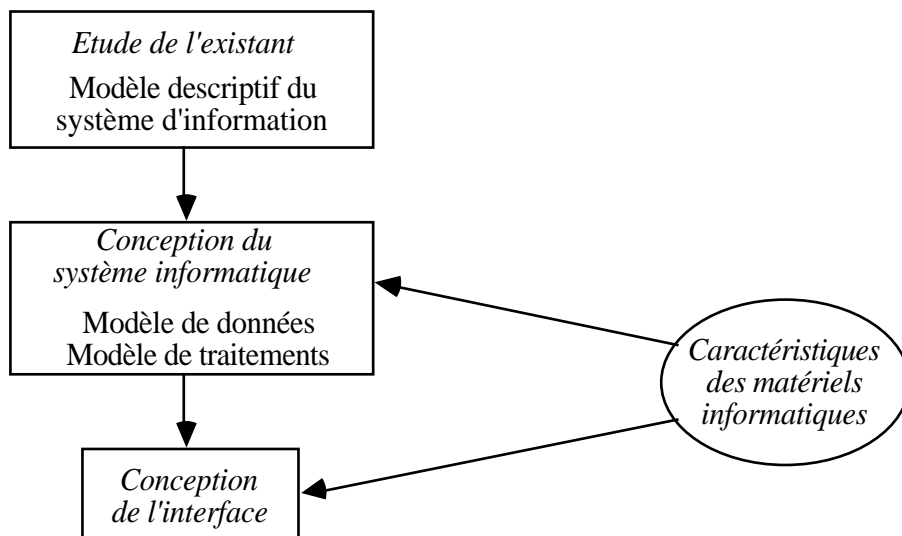
A propos de ce dessin d'écran, on peut se poser les questions suivantes (questions toujours centrées sur l'utilisateur):

- "Toutes les zones de l'écran sont-elles pertinentes pour l'utilisateur?"

- "Le vocabulaire des données ou des commandes correspond-il au vocabulaire employé par les utilisateurs?"

- "La présentation de l'écran est-elle compatible avec les formulaires existants manipulés par les utilisateurs?"
- "Si l'enchaînement des zones à remplir est automatique, correspond-il à une logique d'utilisation et à ces variabilités?"

En résumé, nous pouvons dire que les méthodes d'analyse n'incitent pas l'analyste à compléter sa conception par le point de vue de l'utilisateur. Elles l'incitent au contraire à déduire sa conception du dialogue homme-machine de la logique du système informatique, comme nous l'indiquons sur le schéma suivant:



Nous allons montrer dans la suite de ce livre quelles sont les modifications à apporter aux méthodes de conception de manière à pouvoir intégrer le point de vue de l'utilisateur.

Nous insistons beaucoup sur les méthodes car nous pensons qu'elles sont à l'origine de la plupart des interfaces inadaptées aux utilisateurs alors que le contexte actuel du matériel et du logiciel de base y est favorable .

En effet, le matériel informatique évolue vers de plus grandes capacités en mémoire centrale pour un coût plus faible. Cette tendance est fondamentalement favorable au développement de logiciels adaptés aux utilisateurs car de tels logiciels nécessitent, comme nous le verrons, plus d'espace mémoire pour chaque utilisateur. Chaque nouvelle possibilité offerte à l'utilisateur se traduit par une augmentation de la place mémoire nécessaire.

Sur le plan des logiciels de base, nous voyons l'apparition de logiciels de gestion d'écran plus sophistiqués (voir chapitre 4) qui offrent plus de possibilités d'interaction à l'utilisateur tout en déchargeant le programmeur d'une partie de la gestion du dialogue.

L'évolution du matériel informatique et des logiciels de base est donc tout à fait favorable à l'émergence de logiciels de plus en plus adaptés aux utilisateurs mais paradoxalement ce sont les méthodes d'analyse et de programmation qui constituent à l'heure actuelle un frein à cette évolution; c'est cette lacune que ce livre tente de combler.

1.1.2 Les grands mythes

Dans ce paragraphe, nous avons regroupé un certain nombre de propos que l'on peut entendre et qui sont, à notre avis, des mythes qui entraînent des confusions dans la conception de la communication homme-machine.

Interface et application conversationnelle

Une application conversationnelle a en commun avec une application différée le fait de définir une structure de données et une structure de traitements associée. A priori, la seule grande différence est la conception d'une interface homme-ordinateur qui permette la communication entre l'utilisateur et l'ensemble des fonctionnalités de l'application (traitement et données).

Partant de cette constatation, il semble logique de déduire que, si on veut améliorer la conception des applications interactives, il faut faire porter son effort sur la conception de l'interface indépendamment des fonctionnalités du système qui peuvent être définies par ailleurs par les méthodes d'analyse classiques.

L'indépendance de l'interface semble être confirmée par les propos des concepteurs de "logiciels de base d'interface évoluée" (voir chapitre 4) qui basent leurs travaux sur la séparation entre les fonctionnalités de l'application et son interface utilisateur.

Mais, comme nous le développons dans ce livre, nous verrons qu'une réflexion plus approfondie sur l'amélioration de la communication homme-machine rend impossible d'isoler conceptuellement l'interface et les fonctionnalités car cette amélioration passe par une modification de l'interface mais aussi par une reformulation de la structuration des traitements.

En d'autres termes, la structuration des traitements pour les applications conversationnelles et les applications différées ne peut être identique car, dans le cas d'applications différées, on ne doit prévoir que des enchaînements automatiques entièrement pilotés par l'ordinateur alors que dans le cas d'applications conversationnelles on doit tenir compte d'un partage de pilotage entre l'homme et l'ordinateur.

En fait, ce propos n'est pas en contradiction réelle avec ceux des concepteurs de logiciels de base car il ne se situe pas au même niveau de préoccupation.

Nous disons qu'il y a une *interdépendance conceptuelle* entre interface et fonctionnalités mais il y a une *indépendance physique* nécessaire lors de l'implémentation essentiellement pour des raisons de diminution de complexité et de modifiabilité.

Communication homme-machine

L'expression "homme-machine" ou "utilisateur-ordinateur" peut présenter une ambiguïté qu'il est nécessaire de lever vu sa fréquence d'utilisation dans ce livre.

En cours d'exécution d'une application interactive, il y a en effet une communication entre un homme et un ordinateur, mais de fait lors de la conception c'est l'analyste et le programmeur qui ont emmagasiné sous la forme d'un logiciel un savoir sur l'application.

L'utilisateur dialogue avec ce logiciel d'application et donc avec le concepteur de ce logiciel.

Cette remarque a l'avantage de montrer clairement que l'amélioration de la communication homme-machine, en l'absence "d'améliorations" de l'homme, ne peut porter que sur l'amélioration de la conception.

Langage naturel

Il existe une idée très répandue selon laquelle la communication entre l'homme et la machine serait optimale si l'ordinateur comprenait le langage naturel. Et qu'il suffit donc d'attendre que les recherches dans ce domaine aient abouti.

Sans nier l'intérêt d'une communication en langage naturel, ce type de communication ne peut à lui seul résoudre toutes les difficultés.

Et ceci pour deux raisons essentielles:

- dans le cas d'utilisateurs professionnels, il se crée un vocabulaire spécialisé, laconique, adapté à la tâche (voir chapitre 2) qui utilise un sous-ensemble restreint et parfois déformé du langage naturel.
- dans tous les cas, le langage ne constitue qu'une partie de l'interface (voir chapitre 3) qui n'est elle-même qu'une partie de l'application et ne peut donc en aucun cas suppléer aux autres éléments de l'application.

Intelligence artificielle

La même idée est énoncée au sujet de l'intelligence artificielle et en particulier des systèmes-experts: " il n'y aura plus de problèmes de communication homme-machine quand l'intelligence artificielle sera généralisée".

Nous tenons à faire remarquer que, si les systèmes experts constituent un apport important pour la conception et la réalisation de systèmes d'aide à la décision, ils ne sont pas efficaces pour les autres applications d'informatique de gestion.

En d'autres termes, les systèmes-experts sont adaptés au cas où les processus cognitifs des utilisateurs sont "dirigés par les données" et non par des procédures (comme c'est le cas pour les applications de gestion qui ne relèvent pas de l'aide à la décision).

Par contre, de nombreux concepts présentés ici peuvent servir pour le développement de systèmes-experts plus adaptés aux utilisateurs.

1.1.3 L'apport d'autres disciplines

Pour pouvoir intégrer explicitement l'utilisateur à la conception des applications interactives, il est nécessaire de connaître les caractéristiques des utilisateurs. Pour ce faire, il est nécessaire de puiser dans d'autres disciplines qui ont pour objectif la connaissance de ces utilisateurs.

Deux disciplines nous ont paru particulièrement adaptées pour enrichir notre connaissance de l'utilisateur dans le contexte du dialogue homme-machine; il s'agit de la psychologie cognitive et de l'ergonomie que nous définissons ici.

Psychologie cognitive

La *psychologie cognitive* désigne la partie de la psychologie qui s'intéresse à la manière dont l'homme traite l'information (mémorisation, apprentissage,...). Les concepts qui y sont développés ne sont pas du tout spécifiques de l'interaction homme-machine mais il est évident aussi qu'ils entraînent des conséquences sur cette interaction.

En effet, dans le dialogue homme-machine, c'est essentiellement la capacité de l'homme à traiter des informations symboliques qui est mise en jeu.

Aussi, certains des concepts de la psychologie cognitive, nous ont paru indispensable pour comprendre les réactions des utilisateurs et nous en avons tiré des conséquences sur la conception des applications interactives.

Ergonomie

L'ergonomie s'intéresse de manière générale à l'amélioration des conditions de travail, et l'ergonomie du logiciel concentre plus particulièrement son attention sur les conditions de travail liées à l'utilisation par l'homme de logiciels interactifs.

Le terme "ergonomie du logiciel" a émergé dans les années 80 quand on a commencé à se poser sérieusement le problème de la qualité de l'interaction homme-machine.

On peut remarquer qu'on ne parle pas de l'ergonomie de l'informatique car on distingue toujours l'ergonomie du matériel de celle du logiciel.

Les travaux sur l'ergonomie du matériel sont antérieurs à ceux sur l'ergonomie du logiciel et ont déjà été largement publiés.

En conséquence ils ne sont pas repris dans ce livre; nous citons en bibliographie un livre complet sur ce sujet de Cackir, Hart et Stewart (Cac,80).

Les limites de cette approche

Par définition, l'ergonomie du logiciel ne s'intéresse qu'à un sous-ensemble particulier des conditions de travail, l'amélioration du dialogue homme-ordinateur ne suffit pas à améliorer globalement les conditions de travail; de plus, il peut y avoir un aspect arbitraire et réducteur à s'occuper exclusivement d'ergonomie du logiciel en négligeant d'autres aspects au moins aussi importants au yeux des utilisateurs.

Pour s'en convaincre, nous faisons référence au livre publié par ACTIF (Act,81) qui traite de l'ensemble des modifications des conditions de travail liées à l'informatisation.

ACTIF regroupe les variables concernant l'impact de l'informatique sur les conditions de travail en quatre rubriques:

- *l'emploi*, qui recouvre des variables quantitatives comme le nombre d'emploi ou la durée du travail et des variables qualitatives liées à l'évolution du contenu du travail et de la qualification correspondante
- *l'organisation*, qui désigne d'une part la répartition globale du travail entre les différents partenaires de l'entreprise, d'autre part la description du travail au niveau individuel (type d'activité, répartition des tâches entre l'homme et l'ordinateur,...)
- *les conditions matérielles*, qui désignent les conditions générales comme le bruit, le rythme de travail, etc...et les conditions spécifiques au travail sur terminal (lisibilité, surcharge, dispositifs d'entrée de données,...).
- *les relations de travail*, c'est-à-dire la nature des relations qui s'établissent entre les différents membres de l'entreprise et avec les personnes extérieures.

Le tableau suivant résume l'impact des différentes étapes de l'analyse informatique sur les différentes rubriques décrivant les conditions de travail:

Rubriques liées aux conditions de travail

<i>Etapes d'analyse</i>	Emploi	Organisation	Conditions matérielles	Relations de travail
Schéma Directeur				
Etude préalable				
Etude détaillée				
Réalisation, maintenance				

- Interaction très importante
- Interaction importante
- Interaction peu importante
- Non examiné

Nous pouvons constater que ce que l'on désigne sous le nom d'ergonomie du logiciel regroupe en fait une partie de la rubrique "organisation du travail" au niveau individuel et une partie de la rubrique "conditions matérielles" au niveau informatique.

De plus, les effets des différentes rubriques ne sont pas indépendants et certains éléments peuvent prendre à un moment donné une importance accrue par rapport aux autres; par exemple, une menace sur l'emploi ou de mauvaises relations de travail peuvent gommer complètement l'effet bénéfique d'un logiciel "ergonomique". Inversement de bonnes relations de travail peuvent minimiser l'effet de mauvaises conditions matérielles.

Par ces remarques, nous voulons simplement préciser que la conception d'un logiciel adapté aux utilisateurs n'est pas en mesure de résoudre l'ensemble des problèmes humains posés par l'introduction de l'informatique mais qu'inversement, il ne peut pas y avoir un processus d'informatisation réussi sans optimisation de la manière dont l'homme est amené à communiquer avec l'ordinateur.

1.1.4 Les différents points de vue

Les applications interactives sont conçues, étudiées et utilisées par des personnes différentes ayant des objectifs différents.

Nous pouvons citer au moins trois types de personnes qui s'intéressent de par leur travail à ces logiciels; ce sont l'utilisateur, l'informaticien et l'ergonome. On pourrait citer d'autres intervenants comme le conseiller en organisation ou le formateur, mais dans ce livre, nous nous concentrons sur le point de vue des utilisateurs, des informaticiens et des ergonomes.

Parce que l'objectif de chacune de ces personnes est différent, leurs perceptions de l'interaction homme-machine sont différentes et, en particulier, leurs décompositions de l'interaction et leurs critères d'optimisation différent. Nous nous trouvons devant le problème classique, en théorie des systèmes, des *vues multiples sur un même objet*..

Nous décrivons ci-dessous succinctement ces différents points de vue.

L'objectif de *l'utilisateur* est d'avoir un logiciel lui permettant d'exécuter sa tâche avec le plus de facilité possible.

Il existe peu d'études sur la perception des utilisateurs; mais, nous avons trouvé dans une étude réalisée par Rolloy (Rol,82) un découpage selon les paramètres suivants:

- lisibilité des sorties
- adaptation du logiciel aux situations réelles de travail
- aide à l'utilisation et à l'apprentissage
- opérations redondantes
- initiatives possibles pour l'utilisateur.

L'ergonome a pour but d'améliorer la communication homme-machine et, pour ce faire, il observe le couple homme-machine de l'extérieur.

Le découpage qu'il adopte en est une conséquence et on trouve dans la littérature en ergonomie un consensus sur un découpage en sept paramètres:

- le séquençage des opérations
- le langage d'interaction
- les dispositifs d'entrée
- les dispositifs de présentation
- le temps de réponse
- le traitement des erreurs
- le guidage.

L'objectif de *l'informaticien* est double selon la phase de conception où il se trouve:

- en phase d'analyse son objectif est de définir un ensemble de spécifications qui traduisent sur le plan informatique les fonctionnalités de l'application;
- en phase de programmation, son objectif est de produire un logiciel fiable compte-tenu des spécifications fonctionnelles et des contraintes matérielles et logicielles.

En phase d'analyse et avec des variantes selon les méthodes, on définit des paramètres tels que procédure, opération, synchronisation, entité, relation,...

En phase de programmation on rajoute des paramètres tels que transaction, type, rubrique,...

La correspondance entre les différents points de vue

La première remarque que l'on peut faire est que tous ces paramètres ne se correspondent pas de façon biunivoque, comme nous le montrons sur le tableau suivant où l'on a mis en correspondance les paramètres perçus par l'utilisateur et ceux utilisés par l'ergonome:

<i>Point de vue de l'utilisateur</i>	<i>Point de vue de l'ergonome</i>						
	Séquence-ment des opérations	Langage d'interrogation	Dispositifs d'entrée	Dispositifs de sortie	Temps de réponse	Traitement des erreurs	Guidage
Lisibilité		*		*			*
Adaptation aux situations réelles	*	*	*	*	*	*	
Aide	*		*			*	*
Redondance	*	*	*	*		*	
Initiative	*						

** indique la correspondance des points de vue*

Par la suite, nous nous préoccupons plutôt de la correspondance entre le point de vue de l'ergonome et celui de l'informaticien afin de pouvoir intégrer à la conception informatique les connaissances acquises par les ergonomes.

Pour établir cette correspondance deux approches sont possibles:

- la première approche est abstraite et se base sur la linguistique pour trouver un cadre théorique commun; cette approche reste pour le moment du domaine de la recherche;

- la deuxième approche est plus directement opérationnelle et consiste à traduire chaque paramètre provenant de l'ergonomie selon la logique de conception utilisée en informatique.

C'est cette deuxième approche qui a été choisie et qui est développée dans les chapitres 2 et 3.

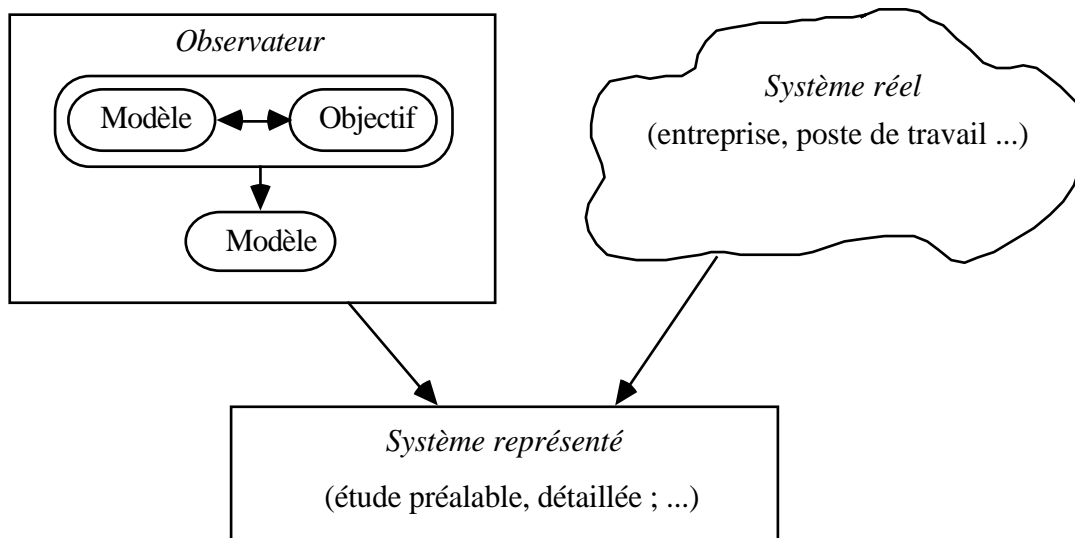
1.2 Les principaux concepts développés

1.2.1 Les concepts classiques

Avant de faire une présentation générale de l'ouvrage, il nous semble important de définir ce que nous entendons par modèle, formalisme, méthode et outil dans le contexte de l'interaction homme-machine.

Un modèle est un ensemble de concepts au moyen desquels nous pensons pouvoir décrire de manière opérationnelle un système réel. Le système représenté au moyen de ce modèle ne peut en aucun cas être confondu avec le système réel qui est dans ce cas le système homme-machine.

En effet, l'approche de plus en plus répandue de la théorie des systèmes, nous a fait prendre conscience du fait qu'une description exhaustive d'un système réel est tout à fait illusoire et que la description d'un système réel est dépendante de deux éléments qui sont l'objectif et le modèle de l'observateur, ce que nous représentons sur le schéma suivant:



Le choix du modèle par l'observateur dépend de l'objectif qu'il se fixe quand il observe le réel; c'est-à-dire qu'il ne choisira pas les mêmes éléments de description selon l'utilisation qu'il veut faire des résultats de son observation.

L'observateur est, dans ce contexte, l'analyste, l'ergonome ou toute personne intervenant dans le processus de conception.

Le système représenté (qui est décrit à des niveaux de détails différents dans les différents dossiers d'analyse) est volontairement un système appauvri du système réel en ce sens qu'il contient moins d'éléments et moins d'interrelations entre ces éléments que le système réel.

Ce fait n'est pas handicapant si les éléments du modèle sont choisis avec soin de manière à permettre de réaliser les objectifs fixés.

C'est pour cela qu'il est nécessaire d'intégrer explicitement, au modèle des applications interactives, des éléments décrivant l'utilisateur, sinon le système représenté n'en contiendra pas.

Le modèle des applications interactives que nous proposons est décrit dans les chapitres 2 et 3.

Le formalisme est une convention de représentation des concepts du modèle. Par exemple un schéma de base de données peut être représenté selon un formalisme graphique ou un formalisme littéral et linéaire.

Le choix du formalisme se fait en fonction de sa lisibilité pour l'observateur et en fonction de propriétés qui peuvent en être déduites (c'est le cas des formalismes mathématiques).

Nous proposons dans le chapitre 2 un formalisme graphique permettant de représenter les éléments du modèle.

La méthode constitue un mode d'emploi particulier d'un modèle; elle indique dans quel ordre observer les différents éléments et comment procéder à ces observations.

Elle restreint ainsi les possibilités du modèle mais elle a l'énorme avantage de permettre un apprentissage plus aisé.

Par contre quand le modèle a été bien intériorisé et que l'analyste a l'habitude des observations sur le terrain, il peut abandonner en partie ou totalement la méthode, de manière à disposer d'une plus grande latitude décisionnelle lui permettant de mieux s'adapter aux spécificités du système réel qu'il observe.

La méthode de conception des applications interactives adaptée aux utilisateurs est décrite au chapitre 5.

A titre d'exemple, nous pouvons citer MERISE qui est présentée d'abord par son modèle puis par la méthode associée alors que AXIAL est présentée uniquement sous forme de méthode.

Un "outil" est un logiciel d'aide à l'analyse qui utilise le formalisme et la méthode associée afin d'augmenter la productivité de l'analyse par des gains de temps d'analyse et de programmation et des gains de fiabilité par des vérifications de cohérence et des simulations.

Dans le cadre de ce livre, nous présentons un modèle d'application interactive adaptée aux utilisateurs ainsi qu'un formalisme et une méthode associée. Cet ensemble peut, bien sûr, être utilisé sans

"outil" associé; aussi le problème de l'outil qui, à lui seul, constitue un développement important n'est pas traité dans ce livre.

1.2.2 Les nouveaux concepts sur les applications interactives et les utilisateurs

Intégration des différents points de vue

Pour pouvoir intégrer explicitement le point de vue de l'utilisateur à la conception des applications interactives, nous proposons un modèle qui comprend trois points de vue, celui de l'analyste, celui de l'utilisateur, celui du programmeur.

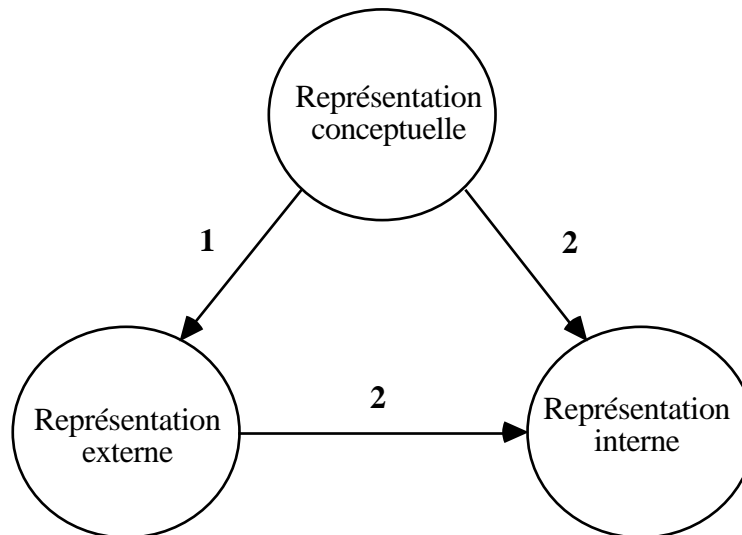
Le point de vue de l'analyste, appelé Représentation Conceptuelle, décrit d'abord la logique générale du nouveau système d'information puis la logique des traitements par rapport à un poste de travail; c'est dans cette deuxième phase que nous commençons à intégrer des éléments de psychologie cognitive liés à l'utilisateur.

Le point de vue de l'utilisateur, appelé Représentation Externe, correspond au logiciel tel qu'il sera vu et manipulé par l'utilisateur. Cette Représentation Externe comprend deux parties; la première est la traduction des éléments de la Représentation Conceptuelle qui entrent dans l'interaction homme-machine; la deuxième comprend la définition de tous les éléments spécifiques à la Représentation Externe (qui correspond à la notion d'interface). C'est ici que nous intégrons tous les éléments d'ergonomie.

Le point de vue du programmeur, appelé Représentation Interne, correspond à la mise en oeuvre des Représentations Conceptuelle et Externe.

Cette phase ne s'accompagne d'aucune spécification nouvelle concernant l'utilisateur et n'intègre donc aucun élément d'ergonomie supplémentaire.

L'ordre de présentation que nous venons de faire correspond à l'ordre usuel de conception comme nous l'indiquons dans le schéma ci-dessous:



On peut modifier cet ordre, en partant d'abord de la Représentation Externe dans le cas de diagnostic d'un logiciel existant ou dans le cas de conception d'un logiciel interactif individuel.

Ces "trois points de vue" peuvent être mis en oeuvre par une même personne, à condition qu'elle se mette effectivement à concevoir l'application avec des points de vue multiples.

L'apport de la psychologie cognitive et de l'ergonomie

Afin d'améliorer la communication homme-machine, il est nécessaire de recueillir le maximum d'information concernant le traitement de l'information en général par l'homme et plus particulièrement dans son dialogue avec un ordinateur.

Pour ce faire, nous avons rassemblé des concepts issus de la psychologie cognitive et de l'ergonomie; cet ensemble de concepts ne constitue en aucun cas une liste exhaustive de l'ensemble des concepts que l'on peut trouver dans ces deux disciplines, mais un sous-ensemble qui nous a paru particulièrement opérationnel pour la conception d'applications interactives adaptées aux utilisateurs.

En ce qui concerne la *psychologie cognitive*, nous avons retenu les concepts suivants:

- il existe différents *types d'utilisateurs* qui, pour des raisons variées, n'utilisent pas le même logiciel de la même manière
- il y a une différence fondamentale entre la *logique de fonctionnement* de l'ordinateur et la *logique d'utilisation* par l'homme de cet ordinateur
- on doit distinguer les *tâches prévues* et les tâches effectivement mises en oeuvre par les utilisateurs
- on peut faire l'hypothèse que la résolution d'une tâche par un homme s'organise selon le modèle de la *planification hiérarchique*

- les caractéristiques de la *mémoire à court terme* chez l'homme ont un impact direct sur le dialogue homme-machine
- une partie de l'apprentissage passe par la création d'*automatismes*..

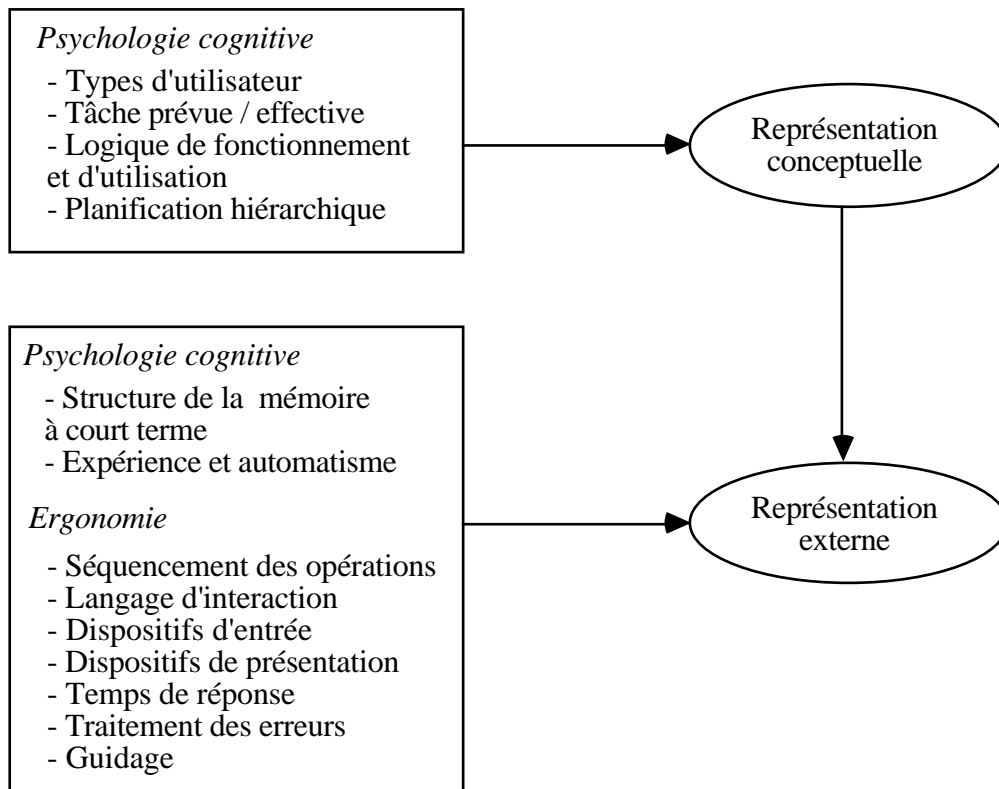
Les quatre premiers concepts sont développés dans le chapitre 2 car ils ont un impact direct sur la conception de la Représentation conceptuelle; les deux derniers sont développés dans le chapitre 3 car ils sont utilisés plutôt pour la conception de la Représentation Externe.

L'ergonomie est l'étude de l'homme en condition de travail en général et une partie des études porte sur les conditions de travail de l'homme face à un ordinateur.

En ce qui concerne l'interaction homme-machine, les études portent sur les paramètres suivants que nous développons au chapitre 3 car ils ont un impact direct sur la conception de la Représentation Externe:

- le *séquençement des opérations* définit l'enchaînement des opérations tel qu'il est autorisé par le logiciel; l'enchaînement peut être totalement libre ou guidé par l'arborescence d'un menu
- le *langage d'interaction* concerne les commandes et les données échangées entre l'homme et l'ordinateur aux niveaux lexical et syntaxique
- les *dispositifs d'entrée* définissent les possibilités physiques d'entrée des données et des commandes dont dispose l'utilisateur
- les *dispositifs de sortie* désignent les possibilités d'affichage des informations, qui dépendent à la fois du matériel et du logiciel
- le *temps de réponse* perçu par l'utilisateur est étroitement dépendant de l'organisation des données et des programmes
- le *traitement des erreurs* désigne les possibilités offertes à l'utilisateur pour détecter et corriger ses erreurs
- le *guidage* définit les aides à l'utilisation et à l'apprentissage incluses dans le logiciel.

Les interactions entre les Représentations Conceptuelle et Externe et les concepts issus de la psychologie cognitive et de l'ergonomie sont représentés dans la figure suivante:



La conception de la Représentation Externe doit intégrer toutes les caractéristiques de l'utilisateur; certains éléments de psychologie cognitive et d'ergonomie sont intégrés directement et d'autres par le biais de la Représentation Conceptuelle (qui la précède).

Il faut remarquer également que les concepts de psychologie cognitive et d'ergonomie ne sont pas complètement indépendants; ce qui est normal étant donné que leur objet d'étude est le même avec des objectifs différents.

Ce fait entraîne que des options prises au niveau de la Représentation Conceptuelle ont un impact sur des éléments d'ergonomie, à savoir le séquencement des opérations, le traitement des erreurs et le guidage.

Nous représentons, sur le tableau suivant, par une étoile le moment où pour la première fois dans le processus de conception, on se préoccupe des différents éléments d'ergonomie:

	Séquence- ment des opérations	Langage d'interro- gation	Dispositifs d'entrée	Dispositifs de sortie	Temps de réponse	Traitement des erreurs	Guidage
Représentation interne					*		
Représentation conceptuelle	*	*				*	*
Représentation externe			*	*			

Il est bien entendu que tous ces éléments auront une "image" dans la Représentation Externe.

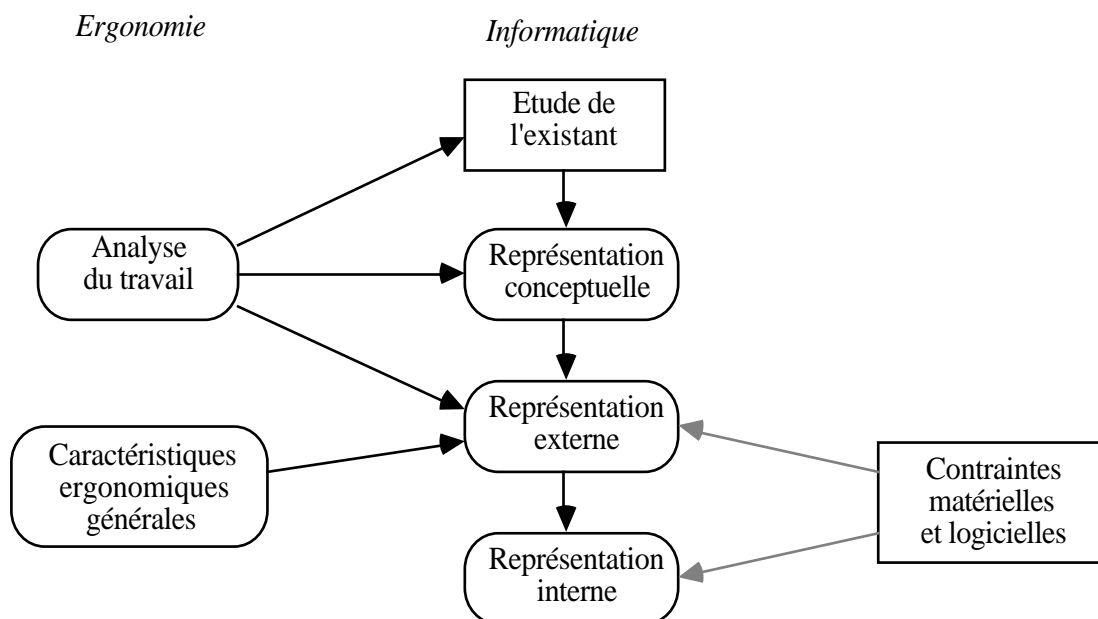
Processus de conception

L'ensemble des caractéristiques des utilisateurs (psychologie cognitive et ergonomie confondues) peuvent être éclatées en deux sous-ensembles disjoints si l'on s'intéresse au processus de conception.

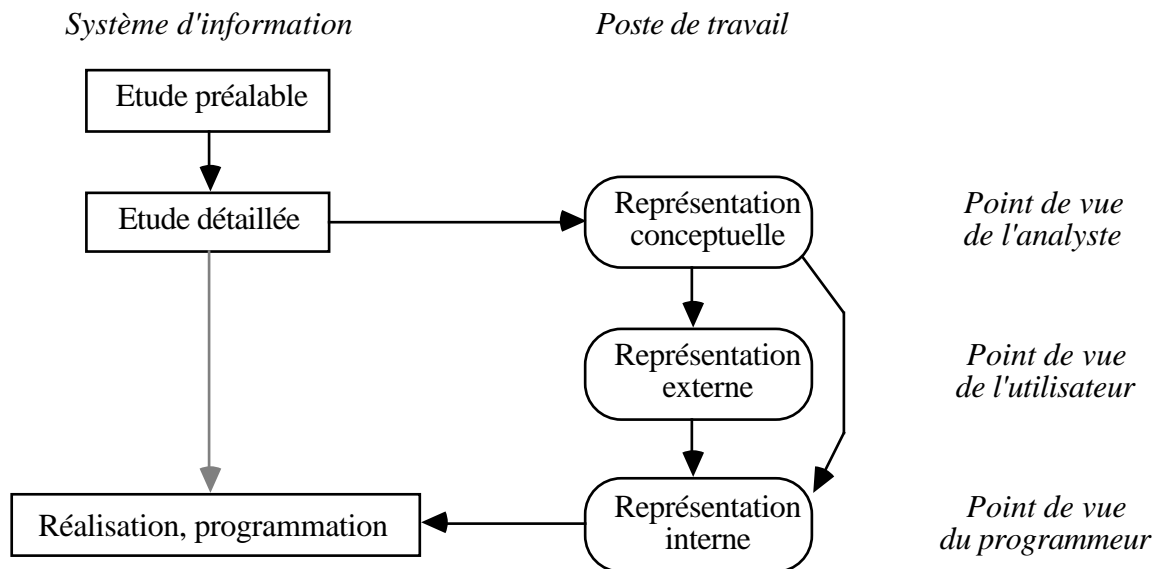
En effet, il est alors opérationnel de distinguer les caractéristiques qui nécessitent une analyse du travail, de celles qui ont un caractère général et qui s'appliquent quelle que soit la situation de travail.

Les éléments recueillis lors de l'analyse du travail ont un impact sur l'étude de l'existant puis sur les Représentations Conceptuelle et Externe.

Nous voyons clairement sur le schéma suivant que la Représentation Externe, c'est-à-dire ce que voit l'utilisateur, n'est plus conçue uniquement à partir de la logique du système informatique:



Nous représentons le lien qui existe entre les méthodes de conception générale des systèmes informatiques et la méthode particulière de conception des applications interactives sur le schéma suivant:



L'on voit que l'ensemble de notre démarche s'applique pour chaque type de poste de travail et peut se greffer sans problème sur une méthode existante.

Le lien avec la méthode MERISE est développé dans le chapitre 5.

1.3 Présentation de l'ouvrage

1.3.1 Composition générale

Nous présentons ici la structure et le contenu des chapitres suivants afin d'indiquer les différentes manières possibles d'aborder cet ouvrage.

Structure

Les chapitres 2, 3 et 4 décrivent l'ensemble du modèle des applications interactives et des caractéristiques de l'utilisateur.

Les chapitres 5 et 6 décrivent deux des méthodes que l'on peut associer au modèle.

Contenu

Le chapitre 2 comprend deux parties; la première expose les concepts de psychologie cognitive qui ont un impact sur la conception de la Représentation Conceptuelle; la deuxième décrit le

modèle de la Représentation Conceptuelle d'un type de poste de travail.

Le chapitre 3 présente d'une part les paramètres d'ergonomie, d'autre part le modèle de la Représentation Externe qui intègre toutes les caractéristiques des utilisateurs (y compris la traduction de la Représentation Conceptuelle).

Le chapitre 4 décrit les conséquences des deux premières représentations sur la mise en oeuvre des logiciels correspondants; il présente également les caractéristiques des systèmes de gestion d'écran de multifenêtrage comme élément de mise en oeuvre.

Le chapitre 5 expose une méthode de conception complète de l'application conversationnelle depuis l'étude de l'existant jusqu'à la spécification de la Représentation Externe; cette méthode est illustrée par une étude de cas.

Le chapitre 6 présente une deuxième méthode qui montre comment faire un diagnostic sur un logiciel existant et comment faire un cahier des charges logiciel respectant des contraintes ergonomiques; la méthode de diagnostic est illustrée par un exemple.

A qui s'adresse ce livre?

Comme nous l'avons dit, ce livre s'adresse à toutes les personnes qui interviennent dans le processus de conception des applications interactives que ce soit les informaticiens- analystes ou les conseillers extérieurs, les correspondants informatiques, les ergonomes, les formateurs.

Il suppose cependant d'avoir des connaissances dans une des méthodes de conception de systèmes informatiques utilisées en informatique des organisations, ou d'avoir pratiqué ce type de conception sur le terrain (sinon certains passages se révéleront obscurs).

1.3.2 Comment le lire?

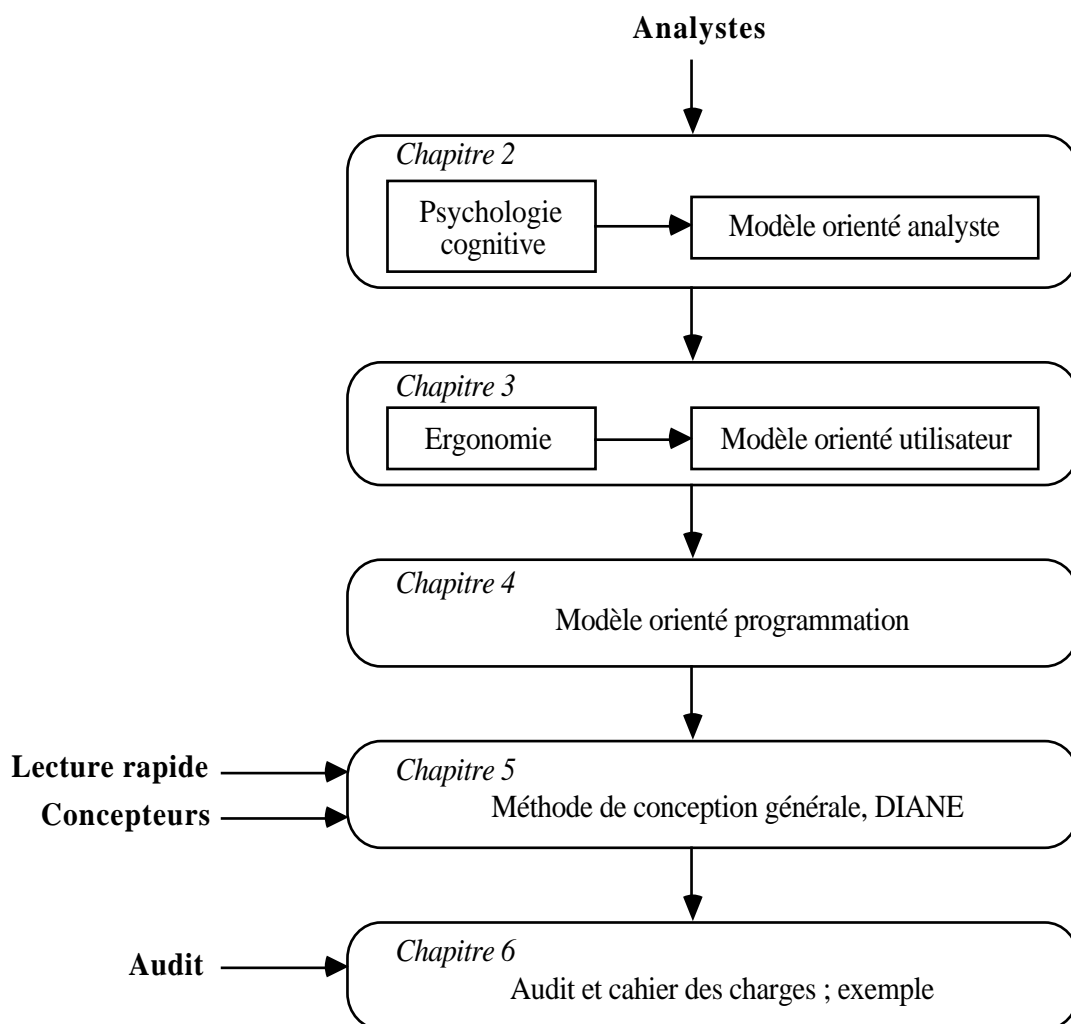
On a l'habitude de lire un livre de manière linéaire; cette habitude est induite par les contraintes techniques d'un livre (contraintes qui n'existent pas dans le cas d'un logiciel), et en conséquence ce livre a été conçu de manière à pouvoir être lu de façon linéaire, mais ce n'est pas la seule manière possible.

Pour aller plus vite, un informaticien-analyste ou un formateur peut prendre directement connaissance de la méthode et des exemples (chapitre 5 et 6) et il peut ultérieurement étayer ces connaissances par la lecture du modèle (chapitres 2, 3, 4).

Un intervenant partiel dans le processus de conception (correspondant informatique, ergonomiste) peut prendre connaissance des concepts de psychologie cognitive et d'ergonomie (première partie des chapitres 2 et 3) puis de la méthode correspondante (chapitre 5).

S'il veut seulement faire un audit d'un logiciel existant, il peut lire en premier la méthode de diagnostic et de conception du cahier des charges (chapitre 6).

Nous représentons les différentes manières possibles de lire ce livre sur le schéma suivant:



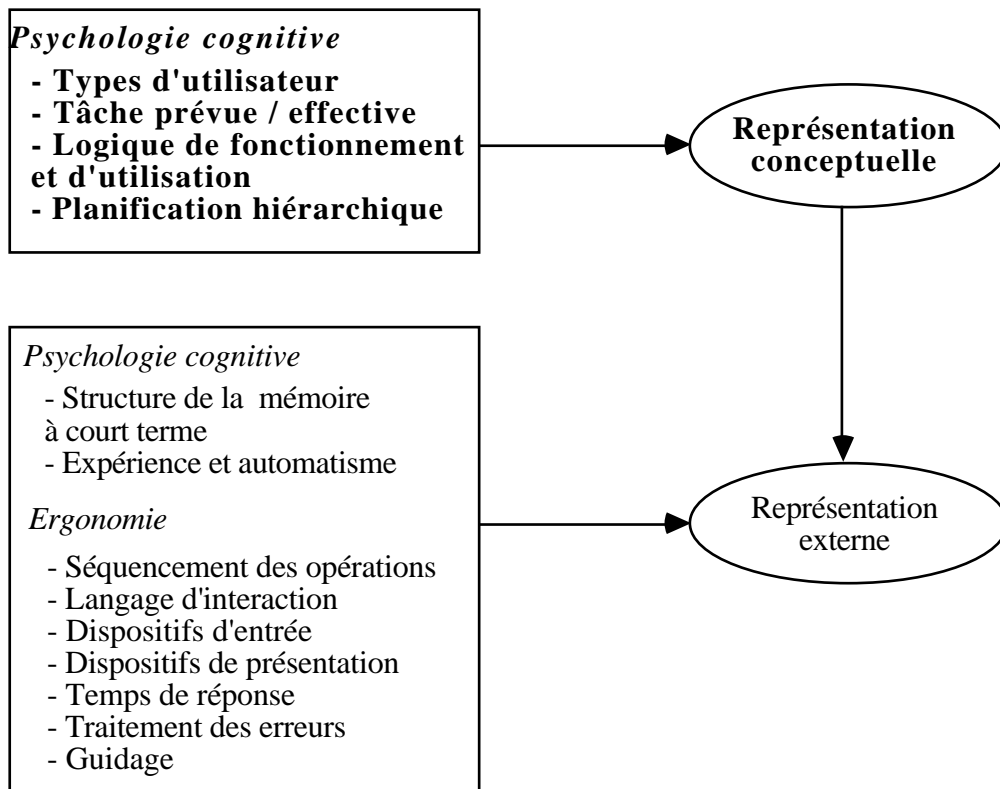
Chapitre 2 :

LES STRUCTURES PROFONDES DES APPLICATIONS INTERACTIVES

Dans ce premier chapitre de présentation du modèle d'applications interactives adaptées aux utilisateurs, nous présentons la Représentation Conceptuelle qui correspond à la spécification de l'application interactive du point de vue de l'analyste.

Comme nous l'avons présenté dans le chapitre 1, la Représentation Conceptuelle intègre certains éléments de psychologie cognitive. Aussi, dans un premier paragraphe, nous présentons les éléments de psychologie cognitive que nous pouvons intégrer à la Représentation Conceptuelle et dans un deuxième paragraphe, nous exposons de manière détaillée le modèle de la Représentation Conceptuelle.

Nous reprenons ci-dessous le schéma présenté au chapitre 1 en indiquant en caractères gras ce qui va être exposé dans ce chapitre.



2.1 La psychologie cognitive et l'utilisateur de logiciel

2.1.1 Eléments pertinents

Notre propos n'est pas ici de faire un exposé général sur la psychologie cognitive mais d'exposer différents concepts qui nous semblent pertinents d'une part pour comprendre les représentations mentales d'un utilisateur de logiciel, d'autre part pour guider l'analyste dans sa conception des applications interactives.

Nous développons ici les concepts suivants :

- l'image opérative qui nous permet de comprendre les images mentales mises en oeuvre dans une tâche spécifique (représentation de tâche)
- la planification hiérarchique qui traite de la décomposition d'une action
- la différence entre utilisateurs novices et expérimentés
- les notions de tâche prévue et tâche effective.

2.1.1.1 L'image opérative

Nous nous référons ici aux concepts développés par Ochanine (Och,72), et repris et approfondis par Sperandio (Spe,80) et Cazamian (Caz, 81).

On appelle "image opérative" les images mentales qui accompagnent l'action de travail. Quand un utilisateur est confronté à un objet dont il reçoit des informations et sur lequel il devra agir, il a une représentation mentale de cet objet.

On appelle structure intégrale l'ensemble des relations existantes entre tous les éléments d'un objet et structure partielle un groupe de relations mis à part pour une raison quelconque.

Pour la réalisation d'une tâche particulière, tous les éléments de la structure intégrale n'offrent pas le même intérêt ; certains éléments peuvent être omis, d'autres au contraire mis en valeur. De manière générale, la réalisation d'une tâche requiert une structure partielle dite structure opérative et la représentation de cette structure par l'utilisateur sera appelée image opérative.

Les images opératives sont un sous-ensemble des images mentales orientées vers l'action. Dans le cas où le but du sujet est simplement de connaître un objet, l'image mentale qu'il aura de cet objet s'approchera ou même s'identifiera à la structure intégrale. Les deux caractères essentiels de cette image opérative sont le laconisme et la déformation fonctionnelle.

- le *laconisme* : l'image étant un moyen d'action, elle ne retient de l'objet que les seules propriétés nécessaires à l'action.

- la *déformation fonctionnelle* : c'est une réplique déformée de l'objet, mais déformée par l'accentuation de ce qui est fonctionnellement important pour une tâche donnée dans un contexte donné.

Pour illustrer ces concepts nous donnons deux exemples :

- le premier exemple concerne la confrontation de deux images opératives de deux personnes différentes. Nous sommes dans une salle de contrôle d'une centrale thermique où un opérateur doit contrôler près d'un millier de paramètres. Pour l'aider à interpréter les signaux, l'ingénieur lui fournit un schéma de fonctionnement global de l'installation. Mais on s'aperçoit que ce schéma ne constitue pas une aide mais un gêne car l'image opérative de l'opérateur est très éloignée de ce schéma. Dans son image opérative les parties très automatisées et très fiables ont quasiment disparu et certaines parties posant fréquemment des problèmes sont surévaluées.

Le schéma global est opératif pour l'ingénieur en tant que constructeur d'usine mais il ne l'est pas pour le technicien qui n'a pas à construire l'usine mais à pallier les incidents de son fonctionnement.

- le deuxième exemple concerne des médecins novices et expérimentés.

Trois groupes de médecins (spécialistes expérimentés, généralistes, novices), sont confrontés à des cas réels de maladie de la thyroïde. Après palpation et examens, on demande à chaque médecin d'exprimer sa représentation mentale de la glande malade par un modelage. Les résultats montrent que les médecins les plus qualifiés produisent des modelages présentant des déformations importantes par rapport à la réalité, en ce sens que les parties atteintes de la glande sont hyper-déformées par le modelage, ce qui correspond à une déformation fonctionnelle permettant d'aboutir au bon diagnostic.

Par contre les médecins débutants ou moins qualifiés produisent des modelages plus conformes au réel mais moins riches en informations pertinentes.

2.1.1.2 La planification hiérarchique

Cette notion a été introduite par des chercheurs en intelligence artificielle Sacerdoti (Sac,74) et Cohen, (Coh,82) et reprise par des chercheurs en science cognitive Dermott (Der,78) et Sebillotte (Seb,83) (Seb, 87).

La question qui se pose ici est de savoir comment l'utilisateur se représente le travail qu'il a à faire pour atteindre son objectif. La

réponse à cette question a un double intérêt en informatique car elle peut permettre à la fois une meilleure utilisation et un meilleur apprentissage si le modèle présenté à l'utilisateur se rapproche le plus possible de sa logique d'utilisation.

Le problème posé ici se distingue de celui de la représentation mentale de l'objet étudié (image mentale ou opérative) en ce sens que l'image mentale correspond à la partie statique (structure des données) et la planification hiérarchique à la partie dynamique (structure des traitements).

L'hypothèse de la planification hiérarchique est que l'utilisateur va élaborer un plan d'action à partir du but qu'il souhaite atteindre.

Cette hypothèse nous semble étayée par les travaux faits par Graesser (Gra,81) sur la mémorisation :

- une information structurellement ordonnée est mieux mémorisée, le rappel se faisant par les structures de plus haut niveau.
- un élément conceptuellement relié à d'autres (déjà mémorisés) est mieux mémorisé qu'un élément isolé.

Ce plan d'action lui permettra de raisonner à des niveaux de détail différents en passant par la définition de sous-buts intermédiaires nécessaires à l'accomplissement du but. Certains de ces sous-buts doivent être nécessairement réalisés, les autres sont optionnels, c'est-à-dire qu'ils ne sont formulés qu'en cours d'exécution de l'action lorsque l'utilisateur se rend compte que certaines conditions requises pour l'exécution d'un sous-but ne sont pas remplies. Ces sous-buts ne font pas partie du schéma initial de l'action, ils sont générés par l'utilisateur en cours d'exécution.

La description d'un sous-but comporte :

- une suite d'actions à réaliser
- l'état final obtenu par la réalisation de la suite d'actions
- les pré-requis qu'il est nécessaire de vérifier avant que l'on puisse exécuter la liste d'actions.

Dans le cadre de la description du travail de bureau, ou plus généralement du travail tertiaire, cette décomposition en sous-buts et pré-requis peut être d'une grande souplesse car elle permet ultérieurement à l'utilisateur de se servir de ces sous-buts dans des contextes différents, *sans imposer une programmation rigide*.

Cette méthode de travail peut servir d'une part à étudier des contextes particuliers de travail pour rendre le plus fidèlement possible la logique d'utilisation de l'opérateur, d'autre part à essayer de trouver des sous-buts invariants d'une tâche à l'autre qui

pourraient servir d'éléments constructifs pour la réalisation de n'importe quel but.

2.1.1.3 Utilisateurs expérimentés et débutants

L'existence d'utilisateurs expérimentés et débutants (ou occasionnels) pose deux types de questions :

- existe-t-il des différences de comportements entre ces deux catégories ?
- comment se fait l'apprentissage, c'est-à-dire le passage d'une catégorie à l'autre ?

Différences de comportement

Nous avons déjà vu qu'il existe des différences au niveau des images opératives, c'est-à-dire que plus l'utilisateur est expérimenté, plus il y a une déformation fonctionnelle de l'image opérative.

On peut noter aussi une différence entre les procédures choisies et les résultats obtenus pour réaliser un même objectif. Pour illustrer cette différence, nous nous référons à une étude publiée par Bisseret (Bis,79) portant sur l'effet de l'expérience sur les contrôleurs aériens.

Face à une situation donnée de positions d'avions, le contrôleur doit décider s'il va y avoir un conflit ou non, c'est-à-dire si deux avions risquent de se rencontrer, auquel cas il doit détourner un des deux. Les mêmes situations ont été exposées à un groupe de débutants et à un groupe de contrôleurs expérimentés.

Les résultats montrent que les débutants sont plus discriminants que les expérimentés; en effet, les débutants classent tous les cas dans une des deux catégories de conflit ou de non-conflit alors que les expérimentés laissent un certain nombre de cas dans le doute, car ils pensent que c'est un cas à surveiller sur lequel ils n'ont pas encore assez d'information. Les expérimentés se montrent ainsi plus prudents même s'ils sont amenés à détourner plus d'avions que les débutants.

Au niveau de l'activité cognitive, il semble que les expérimentés ont un jugement plus rapide fondé sur une catégorisation grossière des situations, alors que les débutants cherchent à calculer pour connaître avec précision la distance séparant deux avions.

Nous n'avons pas d'exemple d'études de différence portant sur le travail de bureau, mais ces résultats ont un caractère assez général pour que l'on puisse en déduire que l'on devrait observer ces différences aussi dans ce secteur. Par contre, l'absence d'études ne nous permet pas de dire sur quels éléments porteraient ces différences.

L'apprentissage

Nous ne voulons pas traiter ici le problème général de l'apprentissage mais seulement celui de l'apprentissage d'un outil comme l'ordinateur. Nous faisons référence ici aux travaux développés par Richard (Ric,83), (Ric,87) qui étudie la différence entre la logique de fonctionnement et la logique d'utilisation.

Dans la *logique de fonctionnement*, on apprend à l'utilisateur le fonctionnement de la machine et les effets de chaque commande du langage de l'ordinateur. "Si P alors Q".

Dans la *logique d'utilisation*, on explique à l'utilisateur "comment faire pour" arriver à un résultat donné. "Si l'objectif est Q alors on peut faire P".

On montre qu'il est difficile de passer d'une logique de fonctionnement à une logique d'utilisation car la procédure à élaborer pour atteindre un but ne peut être déduite directement de la connaissance des règles de fonctionnement ; elle ne peut pas être non plus, dans le cas général, un simple calque de la procédure manuelle.

Pour réaliser à l'aide d'un ordinateur une tâche que l'on sait résoudre par ailleurs, il faut modifier sa représentation du problème en définissant de nouveaux buts et sous-buts compatibles avec le fonctionnement de la machine.

Pour diminuer le fossé entre l'utilisation et le fonctionnement, il est proposé de définir des règles d'utilisation qui précisent les actions qu'il est possible de réaliser et comment le faire.

Les commandes de l'ordinateur ne peuvent, dans le cas général, correspondre à des actions de l'utilisateur car les commandes sont définies par rapport à la sémantique du dispositif et les actions ont un sens pour l'utilisateur dans le contexte des tâches qu'il a à accomplir.

Dans une logique d'utilisation, on pourrait donner l'ensemble des procédures à suivre pour arriver à certains buts mais, dans le cas

général, on ne connaît pas à l'avance l'ensemble des tâches auxquelles peut être utilisé un dispositif, et donc on ne peut élaborer un ensemble exhaustif de procédure.

L'objectif sera donc de définir des actions qui ne correspondent pas à des buts précis mais qui seront des procédures pour la réalisation de ces tâches. c'est-à-dire que l'on aura fait pour l'utilisateur le passage des règles de fonctionnement aux composants d'utilisation.

Dans l'optique de la logique de fonctionnement, le problème qui demeure est de déterminer les bonnes règles d'utilisation à faire apprendre, celles qui correspondent à des actions que le sujet se donne comme objectifs intermédiaires.

2.1.1.4 Tâche et activité

Pour distinguer la tâche de l'activité, nous nous référons à Leplat et Hoc (Lep,83): "Toute analyse de situation dans une perspective psychologique amène à s'interroger sur les rapports entre une tâche et une activité : qu'est-ce qui est demandé au sujet, qu'est-ce qu'il cherche à faire, que fait-il effectivement et comment, et finalement, quels sont les rapports entre ces deux questions ?".

De manière générale, la tâche indique ce qui est à faire, l'activité ce qui se fait.

Pour être plus précis, nous définissons la *tâche* comme la réalisation d'un but donné dans des conditions déterminées.

Le *but* est l'état final. Il peut être décrit par des critères et la valeur qu'ils doivent prendre.

Il existe souvent plusieurs manières équivalentes de décrire le but. Un but peut être décrit et évalué par un procédé qui ne correspond pas à celui mis en jeu pour le réaliser.

Les *conditions* peuvent être décrites de plusieurs manières :

- par l'ensemble des états à parcourir avant d'atteindre l'état final ;
- par les opérations admissibles pour parcourir ces états ;
- par la procédure à mettre en oeuvre pour ce faire, c'est-à-dire la combinaison de ces opérations.

L'*ensemble des états* sera défini par :

- la donnée d'un *état initial* ;
- un découpage temporel du processus en états identifiables ;
- une description des états en valeur de variables qui les caractérisent.

L'*ensemble des opérations* admissibles dans la transformation des états sera appelé le dispositif associé à la tâche.

L'*ensemble des procédures* peut être décrit soit en explicitant la combinaison des opérations sous forme d'algorithme (mode procédural), soit en donnant les propriétés qu'elle doit respecter (mode déclaratif).

La *tâche prévue* est la tâche conçue par celui qui en commande l'exécution.

La description d'une tâche est complète pour un sujet donné quand elle lui permet l'exécution immédiate de la tâche sans nouvelles acquisitions préalables.

Une tâche dont la description est complète ne requiert donc du sujet qu'une activité d'exécution.

Une tâche dont la description est incomplète requiert du sujet une activité d'élaboration en plus de l'activité d'exécution proprement dite.

L'*activité* est ce qui est mis en oeuvre pour exécuter la tâche.

La *tâche effective* correspond à ce que l'utilisateur fait effectivement. La tâche effective n'est pas un décalque de la partie observable de l'activité. Dans la mesure, par exemple, où elle définit une procédure, elle est dérivée de certains comportements et prend aussi en compte ce qu'on sait des règles de fonctionnement du système cognitif. La tâche effective constitue donc un modèle de l'activité. La validation du modèle que constitue la tâche effective se fera par la confrontation des prédictions de ce modèle avec des traces observables de l'activité.

2.1.2 Possibilités d'extrapolation

Nous reprenons dans ce paragraphe les concepts présentés ci-dessus en leur donnant une interprétation utilisable pour la conception des logiciels interactifs.

Du concept d'*image opérative*, nous pouvons retirer trois idées pour les applications interactives:

- la notion de laconisme est importante pour notre conception du vocabulaire et de la syntaxe du dialogue. En effet, certains pensent que la résolution du problème du dialogue passe par l'utilisation et la compréhension du langage naturel. Cette idée peut être vraie pour des utilisateurs occasionnels non spécialisés. Mais dans le cas de spécialistes d'une tâche, il se crée un vocabulaire spécialisé, opératif,

non ambigu qui est fonctionnel et adapté à la tâche. Le rôle de l'analyste est alors de repérer ce vocabulaire et de permettre son utilisation.

- l'image opérative varie selon la fonction des différents utilisateurs. En particulier, l'analyste informaticien et l'utilisateur ont très peu de chance d'avoir la même image opérative des tâches à accomplir. L'analyste a une vision globale et abstraite des procédures de traitement de l'information alors que l'utilisateur a une image locale et concrète de ces mêmes tâches.

- l'image opérative et les processus de décision varient selon le degré d'expérience des utilisateurs. Il faut donc prévoir des présentations différentes du même logiciel pour des utilisateurs de niveau d'expérience différent.

La *planification hiérarchique* correspond à une description du travail orientée-but qui part du résultat que veut atteindre l'utilisateur pour remonter jusqu'aux actions élémentaires nécessaires à la réalisation du but.

Ce type de description n'est pas usuel dans la conception des systèmes d'information où au contraire on fait des descriptions procédurales à partir d'événements d'entrée jusqu'aux événements de sortie. Ces deux démarches ne sont pas simplement inverses l'une de l'autre car le résultat obtenu n'est pas le même:

- dans la description orientée-but, on obtient un ensemble d'opérations nécessaires à la réalisation du but ; des procédures différentes seront regroupées si elles aboutissent au même but ; cette description permet de décrire aisément tous les parallélismes et correspond à la logique d'utilisation.

- dans la description orientée-événement, on obtient l'enchaînement des opérations déclenchées par un ou des événements initiaux ; cette description correspond à la logique de fonctionnement du Système d'Information et décrit les différentes procédures indépendamment de leurs utilisations.

Il n'y a que dans le cas simple où un but est obtenu par la réalisation d'une procédure déclenchée par un événement que ces deux méthodes de description aboutissent au même résultat.

La planification hiérarchique s'applique à notre avis de façon privilégiée pour les raisonnements guidés par les traitements et non pas guidés par les données, c'est-à-dire quand l'enchaînement des opérations est quasi indépendant de la valeur des données ou que cette dépendance ne crée pas une explosion combinatoire de l'arborescence.

Dans l'entreprise, les postes de travail des décideurs correspondent davantage à des modes de travail guidés par les données alors que les postes d'exécution sont plutôt guidés par les traitements.

Pour la conception des applications interactives, la distinction entre *logique de fonctionnement* et *logique d'utilisation* nous semble particulièrement précieuse. En effet, la plupart des logiciels interactifs sont élaborés et présentés à l'utilisateur selon une logique de fonctionnement. Ce résultat peut apparaître normal car, après l'application d'une méthode classique d'analyse, on dispose toujours d'une description fonctionnelle et jamais des logiques d'utilisation. Les logiques d'utilisation doivent être recueillies explicitement lors de l'étude de l'existant et être éventuellement complétées par expérimentation sur prototype. La méthode de recueil des logiques d'utilisation est décrite dans le chapitre quatre.

En ce qui concerne la distinction entre *tâche prévue* et *tâche effective*, on peut se demander quel type de tâche l'on recueille lors d'une étude de l'existant classique?

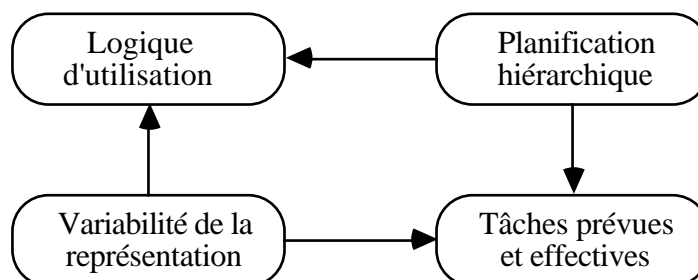
Les méthodes d'analyse ne disent rien à ce sujet; ce qui a pour conséquence qu'en pratique cela dépend de la façon dont on a mené cette étude :

- si on a interrogé essentiellement des responsables, on a beaucoup de chances d'avoir recueilli des tâches prévues.
- si on a interrogé des utilisateurs effectifs, on a plus de chance d'avoir recueilli des tâches effectives ou un mélange de tâches effectives et de tâches prévues.

Pour pouvoir distinguer les différents types de tâches, il faut que l'analyste modifie sa façon de mener les interviews en fonction de cette préoccupation et complète éventuellement son étude par des simulations ou des observations. Pour plus de précision, nous renvoyons également au chapitre quatre.

Interaction entre ces différents concepts

Les différents concepts que nous venons d'exposer ne sont pas indépendants ; nous représentons ces relations sur le schéma suivant :



La planification hiérarchique permet de trouver la logique d'utilisation pour un but donné.

Les variabilités de la représentation se traduiront soit par des logiques d'utilisation différentes, soit par les notions de tâches prévues et de tâches effectives.

Pour décrire les tâches prévues et effectives, on peut utiliser la planification hiérarchique.

2.2 La Représentation Conceptuelle de l'application interactive

2.2.1 Introduction

Les concepts de psychologie cognitive, développés précédemment, concernent d'une part *la représentation des traitements de l'information* de l'utilisateur d'autre part les *variabilités* de cette représentation . Nous résumons ici les concepts que nous avons retenus pour la définition de la Représentation Conceptuelle d'une application interactive.

Représentation du traitement de l'information de l'utilisateur

Nous avons retenu deux notions :

- le concept de *planification hiérarchique* permet de décrire le travail de l'utilisateur en partant des buts qu'il se fixe (ou qui lui sont fixés) et de remonter jusqu'à l'ensemble des opérations et des pré-requis nécessaires à la réalisation de ses buts,
- la différence entre *logique de fonctionnement* et *logique d'utilisation* permet de structurer les menus selon une logique d'utilisation au lieu d'une logique de fonctionnement.

Les variabilités de cette représentation

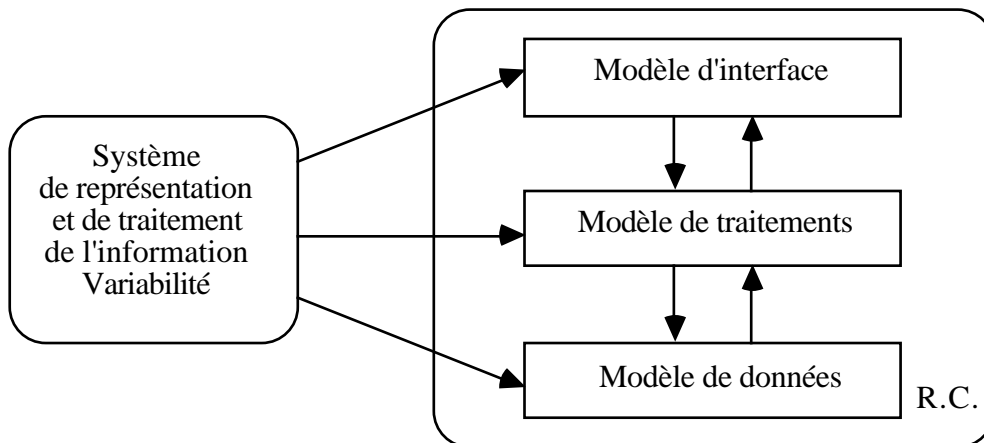
La représentation du traitement de l'information, que nous venons d'évoquer, n'est pas unique pour une tâche donnée ; cette variabilité est due à deux causes :

- la *déformation fonctionnelle* de la tâche qui provient d'une part des différents objectifs de travail des utilisateurs, d'autre part du degré d'expérience professionnelle des différents utilisateurs,

- la différence entre la *tâche prévue* et les *tâches effectives* ; la tâche prévue décrivant la tâche standard telle qu'elle est décrite dans le cas normal ; les tâches effectives décrivant les différentes adaptations de la tâche prévue réalisées par l'utilisateur afin de pouvoir réguler son travail en fonction des aléas de l'environnement.

Intégration à la Représentation Conceptuelle

La Représentation Conceptuelle, qui correspond, comme nous l'avons vu au chapitre 1, au premier niveau de représentation de l'application interactive par l'analyste informaticien, intègre d'une part la description des données des traitements et d'une partie de l'interface (le reste étant défini au niveau de la Représentation Externe), d'autre part la représentation du traitement de l'information de l'utilisateur et la variabilité de ces représentations.



Paramètres constants et variables

Dans l'exposé du modèle de la Représentation Conceptuelle, il est nécessaire de distinguer les *paramètres variables* provenant des caractéristiques particulières d'une application des *paramètres constants* qui sont présents dans toute application interactive.

Les paramètres variables sont issus de l'analyse du travail; il s'agit, par exemple, des buts ou de la logique d'utilisation.

Les paramètres constants sont issus des caractéristiques générales des utilisateurs; il s'agit, par exemple, des possibilités d'interrompre à tout moment ou de transférer des données.

Comme nous le montrons à la fin de ce chapitre, pour des raisons de lisibilité et d'efficacité, seuls les paramètres variables sont représentés dans les schémas décrivant la Représentation Conceptuelle.

2.2.2 Les paramètres variables de la Représentation Conceptuelle

2.2.2.1 Principes de base

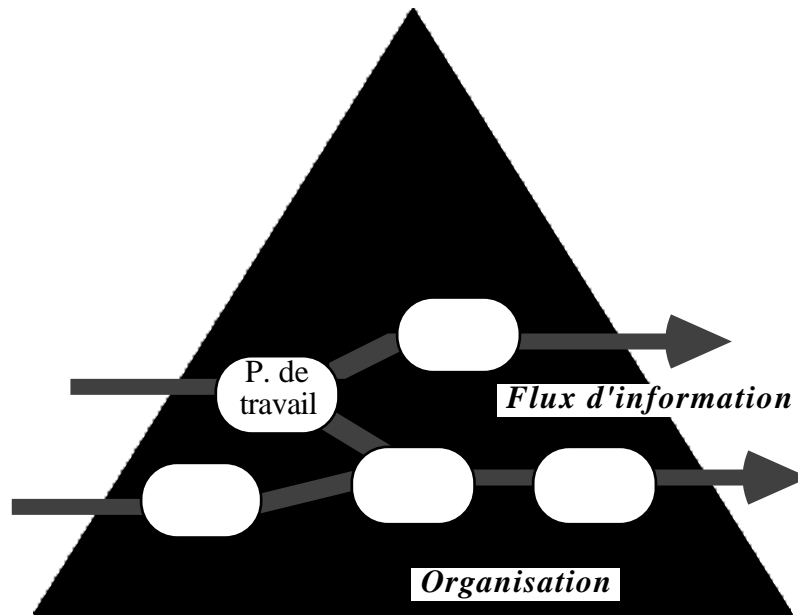
Le poste de travail

Contrairement aux méthodes d'analyse usuelle qui ont des représentations conceptuelles pour chaque fonction du système d'information de l'entreprise, nous aurons une Représentation Conceptuelle pour chaque *type de poste de travail*.

Le poste de travail se situe à l'interaction des grandes fonctions de l'entreprise et de l'organisation de celle-ci. Les fonctions de l'entreprise, qui correspondent aux traitements des flux d'informations qui la traversent, doivent être décomposées en traitements qui correspondent à la quantité de travail que peut effectuer un être humain. A ce découpage se superpose pour des raisons de pilotage et de contrôle de l'entreprise une organisation c'est-à-dire un ensemble de chaînes de pilotage permettant de passer des projets à long terme à une gestion quotidienne (Mel,77).

Un *poste de travail* sera donc décrit par des objectifs ou buts correspondant à la décomposition des fonctions et un niveau de responsabilité correspondant à sa place dans la chaîne de pilotage.

Nous parlerons de *type de poste de travail* pour désigner l'ensemble des postes de travail ayant les mêmes objectifs et les mêmes niveaux de responsabilité.



Nous venons d'insister sur la définition du poste de travail car tous les concepts que nous verrons maintenant se situent par rapport au poste de travail.

Le sens de la description du travail

Généralement, dans les méthodes d'analyse de gestion, la description du travail est de type opérationnel : elle part d'événements déclenchants (ex: un bon de commande arrivant dans l'entreprise) et elle suit le cheminement de cet événement tout au long des transformations qu'il subit jusqu'à la fin du processus. Nous appellerons cette démarche *descendante* dans le sens où elle décrit le système d'information depuis les événements initiaux jusqu'aux événements terminaux.

Dans notre Représentation Conceptuelle, nous avons décidé d'utiliser une démarche inverse qui est plus proche de nos connaissances du fonctionnement des opérateurs.

Dans notre démarche que nous appellerons *ascendante*, nous partirons de la description du (ou des) but(s) du poste de travail pour remonter jusqu'aux événements initiaux. Le *but* se définit à la fois par rapport à l'organisation et par rapport au poste de travail. Il permet d'une part à l'utilisateur de réguler son activité, d'autre part à l'organisation de juger les performances du poste de travail par rapport aux objectifs globaux. Les buts sont définis au minimum pour permettre la survie du système.

La tâche

Nous prendrons la définition de la tâche développée en ergonomie.

La *tâche* est décrite par un *but* (état final) et des *conditions* ; les conditions peuvent être décrites de l'une des trois façons suivantes :

- l'ensemble des *états* à parcourir pour atteindre le but
- les *opérations* admissibles pour parcourir ces états
- la *procédure* ou combinaisons de ces opérations.

L'activité est une occurrence de réalisation d'une tâche ; à une tâche correspondront plusieurs activités.

Dans toute la suite de cet exposé, nous choisissons de décrire les conditions d'une tâche par une *procédure*, parce que c'est l'approche qui convient le mieux à la majeure partie des postes de travail concernés par l'automatisation, c'est-à-dire les postes de responsabilité de niveau moyen ou faible (où les raisonnements sont guidés par les traitements). Pour les postes de haute responsabilité (où les raisonnements sont guidés par les données), une approche Intelligence Artificielle basée sur les opérations admissibles serait plus indiquée.

La procédure

Nous définissons :

- *le but*, comme l'état final que doit atteindre le système homme-machine. Cet état final sera décrit par des critères et la valeur qu'ils doivent prendre, le même but peut être décrit de plusieurs manières équivalentes.
- *la procédure*, comme une combinaison d'opérations permettant d'atteindre le but ; elle sera décrite par:
 - une liste d'opérations
 - une liste de pré-requis, chaque pré-requis exprimant les conditions qui doivent être réalisées pour qu'une opération puisse être effectuée.

Dans le cas général, la notion de pré-requis permettra d'exprimer la notion de *parallélisme*, particulièrement importante dans le cas d'applications interactives pour augmenter les possibilités d'ajustement et de souplesse de l'utilisateur.

Selon leur complexité, les opérations pourront être décomposées en plusieurs niveaux.

La plus petite décomposition possible sera appelée *opération élémentaire* ; elle correspond à la plus petite unité de traitement de

l'information (commande ou donnée) qui ait un sens pour l'opérateur.

Le but, en fonction de sa complexité et de sa nature, pourra lui aussi être décomposé en sous-but, sous-sous-but... jusqu'à ce que l'on arrive à un niveau de traitement de l'information facilement manipulable.

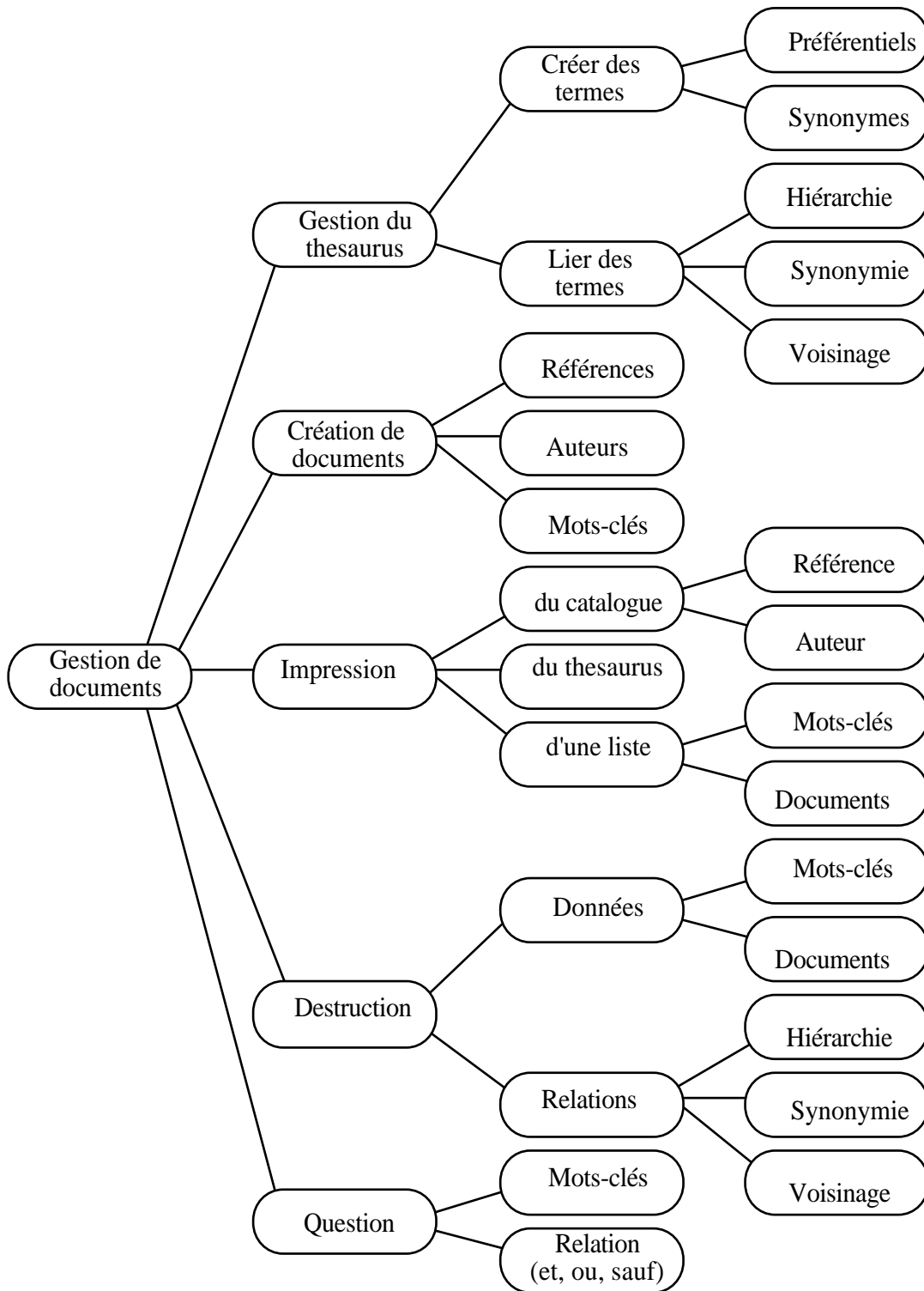
La logique d'utilisation

Les notions de but, opération, pré-requis font référence exclusivement à *la représentation mentale* que l'utilisateur a de son travail (*logique d'utilisation*) ; à aucun moment nous ne faisons référence ici à un découpage correspondant à la logique du traitement informatique ce qui a pour conséquence que nous n'obtiendrons pas forcément le même découpage.

Nous allons illustrer ces différences de découpage des traitements sur un exemple.

Exemple de gestion de documents de bibliothèque

La première décomposition de cette fonction en opérations correspond à un découpage "informatique" (logique de fonctionnement) où le découpage et les regroupements correspondent à la structure des données et aux traitements effectués sur ces données (création, impression, destruction, consultation) . Nous avons un fichier thesaurus et un fichier documents, ce qui donne le découpage suivant :



La deuxième décomposition correspond à une logique d'utilisation et nous avons donc à définir les sous-buts assignés à ce poste de travail concernant le but "gestion des documents".

L'analyse du travail montre que nous avons deux sous-buts :

- enregistrer

- consulter.

Pour "enregistrer", la liste des opérations possibles est la suivante :

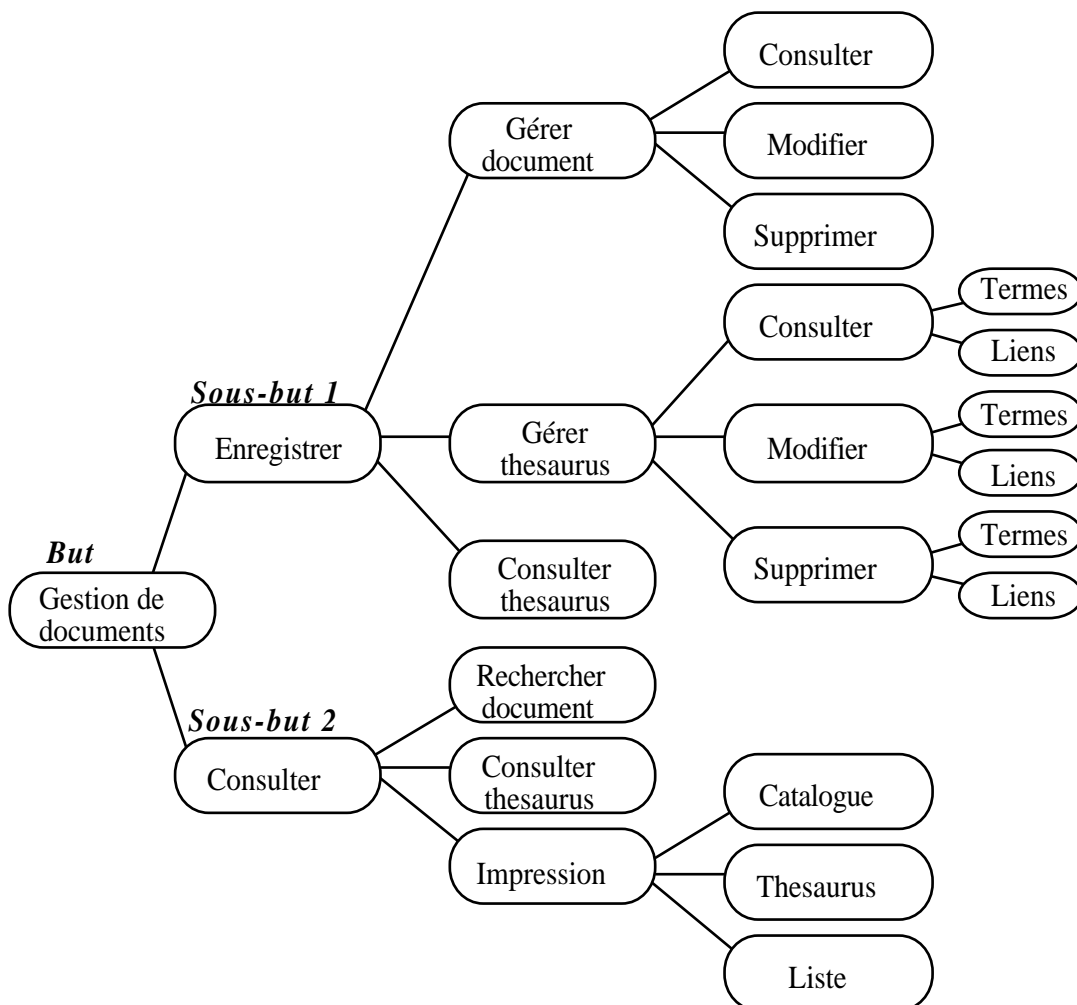
- gérer les documents (création, modification, suppression)
- gérer le thesaurus (création, modification, suppression de mots-clés ou de liens)
- consulter le thesaurus.

Pour "consulter", la liste des opérations possibles est la suivante :

- consulter les documents (recherche à partir de critères ou de mots-clés)
- consulter le thesaurus
- imprimer les résultats.

Il n'y a pas de précédences spécifique à l'application autre que celles implicites liées à la gestion des données (on ne peut modifier, consulter ou détruire une information qui n'a pas été créée).

La représentation visuelle peut se présenter comme suit :



Nous remarquons que l'opération "consulter le thesaurus" figure dans les deux sous-buts, cela signifie que l'opérateur peut avoir besoin de cette opération dans chacun de ces sous-buts et qu'il pourra ainsi y accéder sans *repasser par une arborescence de commandes*.

Note : il resterait à redécouper les sous-buts intermédiaires :

- consulter le thesaurus
- gérer le thesaurus
- gérer les document
- imprimer
- consulter le document.

fin de l'exemple.

De manière générale, *une même opération pourra appartenir à plusieurs buts*, une même opération élémentaire pourra appartenir à plusieurs opérations.

Le critère essentiel est que l'utilisateur ait "sous la main" toutes les opérations dont il peut avoir besoin pour réaliser un but donné (la duplication logique des opérations n'implique aucune duplication physique sur les supports informatiques).

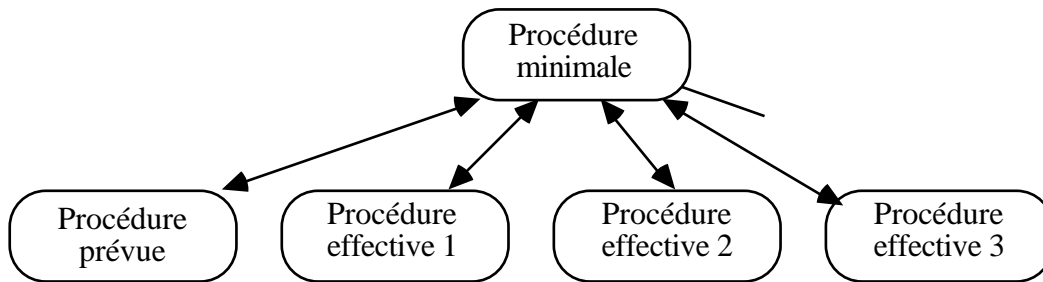
2.2.2.2 Prise en compte de la variabilité dans la R.C

Les procédures multiples

Pour tenir compte de la variabilité d'exécution d'une tâche donnée, nous décrirons plusieurs procédures pour une même tâche, le but restant identique. Nous définissons :

- la *procédure prescrite* (ou prévue) correspondant à la procédure standard ou recommandée qui est la plupart du temps, celle qui est spontanément recueillie par les analystes (interview).
- la *procédure effective* étant celle qui est effectivement mise en oeuvre par un utilisateur donné et qui est souvent recueillie par observation.
- la *procédure minimale* comme l'ensemble des opérations et des enchaînements minimaux nécessaires pour que le but de la tâche puisse être considéré comme atteint par l'ordinateur.

Pour une tâche donnée il y aura *une* procédure minimale, *une* procédure prescrite et *plusieurs* procédures effectives.



Les procédures effectives et la procédure prescrite devront être compatibles avec la procédure minimale.

Cette compatibilité peut s'exprimer de manière formelle mais nous voudrions ici justifier de manière intuitive ces différentes notions et leurs relations avec l'utilisateur.

Le pilotage de l'application

Dans la définition d'une application interactive, on fixe, la plupart du temps implicitement, la répartition du pilotage entre l'homme et l'ordinateur.

Le *pilotage* d'une tâche (ou d'un ensemble de tâches), recouvre deux notions:

- la notion de *contrôle* de l'exécution de la tâche
- la notion de *régulation* qui laisse au pilote la possibilité de modifier certaines conditions d'exécution de la tâche de manière à atteindre l'objectif fixé quels que soient les aléas provenant de l'environnement.

La régulation des tâches relève, dans le cas de l'informatique des organisations, quasi exclusivement de la responsabilité de l'utilisateur. En effet, les caractéristiques propres d'un système d'information d'une organisation sont, d'une part d'être ouvert sur son environnement (ce qui rend très difficile sinon impossible l'établissement d'une liste exhaustive de l'ensemble des informations d'entrées de l'organisation), d'autre part de gérer une très grande variété d'informations d'origines diverses (organisationnelles, psychologiques, sociologiques, techniques) qu'il est souvent difficile d'explicitier et de spécifier. Il s'en suit que les circonstances qui réclament une intervention régulatrice ne peuvent pas être définies formellement et encore moins la nature de ces interrelations.

On appelle *latitude décisionnelle* la part de pilotage qui est sous la responsabilité de l'utilisateur.

Dans le cas général, le contrôle de l'exécution d'une application interactive est partagé implicitement entre l'homme et l'ordinateur selon des proportions très variables qui peuvent évoluer entre deux situations extrêmes qui vont du pilotage entièrement automatisé au pilotage entièrement assumé par l'utilisateur.

Quand le pilotage est laissé entièrement à l'utilisateur, aucun séquençement des écrans n'est imposé à l'utilisateur. Il active les traitements, dans un ordre totalement libre. Le seul contrôle qui est fait par l'ordinateur porte sur l'existence des données manipulées dans les traitements.

Dans ce cas nous laissons de côté trois aspects :

- aucun contrôle n'est effectué sur la cohérence de la procédure par rapport au but fixé
- il n'y a pas de prise en compte du travail de l'utilisateur en tant qu'individu intégré dans une organisation dont le travail a un sens par rapport à cette organisation.
- l'ordinateur n'est pas utilisé au maximum de ses possibilités. On se prive par exemple de la possibilité de faire des enchaînements automatiques ou des inférences.

Ce type de logiciel est en fait très bien adapté pour des postes où les tâches sont peu structurées et qui peuvent travailler de manière relativement isolée par rapport au reste de l'organisation.

Par contre, quand le pilotage est entièrement laissé à l'ordinateur, il y a la garantie d'une cohérence maximale du système d'information de l'entreprise mais au détriment de :

- la souplesse de mise en oeuvre pour l'utilisateur
- la facilité d'intégrer des variabilités et des aléas.

Pour notre part, nous voulons nous situer de manière plus générale par rapport au pilotage ; les deux cas extrêmes que nous venons de citer devenant deux cas particuliers de notre modèle.

Pour cela nous proposons un modèle qui permet d'explicitier et de décrire :

- ce qui doit être *piloté par l'utilisateur*,
- ce qui doit être *piloté par la machine*.

C'est pour cela, que nous avons introduit le concept de procédure minimale qui définit, pour une tâche donnée, les contrôles qui doivent être effectués obligatoirement par l'ordinateur quel que soit le type de procédure prescrite ou effective que l'utilisateur décide

de mettre en oeuvre. La procédure minimale définit, a contrario, les limites de la latitude décisionnelle de l'utilisateur. En effet, tout ce qui ne sera pas explicitement défini au niveau des enchaînements et des déclenchements d'opérations dans la procédure minimale sera considéré comme laissé à la libre disposition de l'utilisateur et ne fera en tout état de cause l'objet d'aucun contrôle systématique de l'ordinateur.

A partir de cette procédure minimale, l'utilisateur peut se définir une procédure prescrite (ou standard) et une ou plusieurs procédures effectives qu'il utilisera à sa convenance et qui le déchargeront dans des cas connus et répertoriés du contrôle de l'exécution de la procédure. Mais il pourra à tout moment "reprenre la main" afin d'imposer son propre pilotage (dans les limites de sa latitude décisionnelle) face à une situation imprévue.

On voit donc que la notion de procédure minimale permet de définir une application interactive ouverte en ce sens qu'il n'est pas utile de définir a priori l'ensemble des procédures effectives mais qu'elles pourront être définies au fur et à mesure des besoins et qu'il nous suffira de vérifier la compatibilité de chaque nouvelle procédure effective avec la procédure minimale.

Pour construire la procédure minimale d'une tâche nous pouvons soit procéder directement si la part de contrôle que l'on doit donner à l'ordinateur est évidente, soit l'extraire de la procédure prévue. Dans ce dernier cas, nous remarquons que la procédure prévue décrit des éléments de nature différentes :

- d'une part les règles de gestion indispensables pour assurer la cohérence et la survie du système.
- d'autre part des règles de commodité, d'usage, dites de "bon sens" qui peuvent être transgressées sans remettre en cause le système mais qui permettent, associées aux règles de cohérence, de décrire une procédure usuelle.

Nous construirons alors la procédure minimale à partir de la procédure prévue dont nous n'aurons retenu que les règles de cohérence correspondant au contrôle que nous souhaitons confier à l'ordinateur.

2.2.2.3 Description détaillée de la R.C

Nous allons définir de manière plus précise les propriétés des opérations et des pré-requis afin de faire le lien avec la structure des traitements et des données. Nous distinguons les propriétés usuelles

de celles permettant de décrire la répartition du pilotage entre l'homme et l'ordinateur.

Propriétés usuelles des opérations

Une opération sera définie comme un ensemble de transactions pouvant être exécutées consécutivement sans attente d'événements externes au système homme-ordinateur (c'est-à-dire que toutes les informations nécessaires au déroulement de l'opération sont soit disponibles en machine soit connues de l'utilisateur). Elle sera décrite par ses entrées, sa nature, son déclenchement, son statut, ses sorties :

- ses *entrées* sont la liste des événements qui peuvent permettre son déclenchement. Un événement sera décrit par un nom et la structure de donnée qui lui est associée.

Un événement initial sera un événement qui n'est pas une sortie d'opération.

Un événement terminal sera un événement qui n'est pas une entrée d'opération.

- sa *nature* sera interactive, automatique ou manuelle :

. une opération est interactive si son déroulement met en oeuvre des actions de l'utilisateur et de l'ordinateur

. une opération est automatique si son déroulement met en oeuvre exclusivement des actions de l'ordinateur

. une opération est manuelle si son déroulement met en oeuvre exclusivement des actions de l'utilisateur.

Par définition, seules les opérations interactives et automatiques sont décrites dans la R.C d'une application interactive. Les opérations manuelles étant décrites au niveau plus global du système d'information existant ou conçu.

- au cours d'une exécution son *statut* est activable ou non :

une opération est activable si les conditions du pré-requis sont réalisées.

- ses *sorties* sont des événements qui pourront déclencher d'autres opérations.

Propriétés usuelles des pré-requis

Les pré-requis recouvrent deux notions distinctes ; d'une part la notion de synchronisation, d'autre part la notion de précondition.

La *synchronisation* est une proposition logique (booléenne) portant sur la liste des événements d'entrée de l'opération O_i
(A .ET. B) .OU. C peut constituer une synchronisation c'est-à-dire que O_i deviendra activable quand soit les événements A et B seront réalisés (ou vrai) soit l'événement C.

A la synchronisation on associera également la notion de délai qui correspond à une règle de temps concernant le déclenchement d'une opération O_i quand celle-ci est devenue activable:

- délai nul : l'exécution de O_i devra avoir lieu dès que O_i est activable
- délai libre : l'exécution de O_i pourra avoir lieu à n'importe quel moment après son activation
- délai avec bornes : O_i devra être déclenché avant une date D_1 et après une date D_2 .

Les *préconditions* expriment une condition sur la valeur des données contenues dans l'événement, conditions qui doivent être vraies pour que l'événement soit considéré comme réalisé.

On peut déduire des pré-requis, la notion de *précédence* entre opérations. La précédence exprime un lien de priorité d'exécution entre opérations. Si $P(O_i, O_j)$ est satisfait cela signifie que O_i doit être exécuté avant O_j .

Propriétés liées au pilotage

Pour préciser le pilotage nous sommes amenés à rajouter des propriétés aux opérations et pré-requis :

- une opération est obligatoire ou facultative :
 - . une opération est *obligatoire* si son exécution, quand elle est activable, est nécessaire pour atteindre le but.
 - . une opération est *facultative* si son exécution ou sa non exécution n'interdit pas d'atteindre le but.
- le *déclenchement* d'une opération est optionnel ou systématique :
 - . le déclenchement de l'opération est *optionnel* s'il dépend de l'utilisateur
 - . le déclenchement de l'opération est *systématique* s'il dépend de l'ordinateur.
- une précédence est *permanente* si elle existe dans toutes les descriptions procédurales de la tâche (procédure minimale, prévue, effective)

- une précedence est *indicative* si elle existe dans une ou plusieurs procédures effectives ou prescrites sans exister dans la procédure minimale.

Il est important de noter que les *propriétés de pilotage sont propres à un but..* Cela veut dire qu'une même opération qui appartiendra à deux buts B_1 et B_2 peut être obligatoire dans B_1 et facultative dans B_2 , et il en sera de même pour les précédences.

La signification de ces propriétés par rapport à la notion de pilotage est la suivante :

- une opération obligatoire est pilotée par l'ordinateur, c'est-à-dire qu'il contrôle son exécution. Le but sera considéré comme atteint par l'ordinateur si toutes les opérations obligatoires activables ont été réalisées.

- une opération facultative est pilotée par l'utilisateur qui en contrôle l'exécution.

Soit en reprenant l'exemple précédent de la gestion de documents;

opérations OBLIGATOIRE :

gérer document, GD

gérer thesaurus, GT

Sous-but 1 (SB_1)

opérations FACULTATIVE :

consulter thesaurus

rechercher document

impression

Sous-but 2 (SB_2)

SB_1 sera considéré comme atteint par l'ordinateur si, pour une occurrence de donnée, GD et GT ont été réalisés. Le But sera considéré comme atteint par l'ordinateur si, pour une occurrence de donnée, SB_1 a été atteint.

Toutes les autres opérations sont laissées à la libre utilisation de l'utilisateur et leur exécution ou non ne fait pas l'objet d'un contrôle par l'ordinateur.

Pour les précédences, nous avons le même type de signification :

- une précedence permanente est pilotée par l'ordinateur qui contrôle ainsi le déclenchement de l'opération associée

- une précedence indicative est pilotée par l'utilisateur qui peut à sa demande être guidé dans le déroulement de la procédure.

Les notions que nous venons de définir sont valables pour toutes les procédures sans distinction de procédure prévue, minimale ou effective.

La procédure minimale exprime le contrôle minimal sur l'utilisateur : elle laisse un pilotage maximal de son travail à l'utilisateur.

Les procédures prévues et effectives sont des variantes possibles de la procédure minimale qui d'une part restreignent le pilotage de l'utilisateur, d'autre part lui facilitent son travail (enchaînements automatique, guidage...).

Les rapports entre propriétés

Nous allons examiner les rapports entre les différentes propriétés. Nous examinerons tout d'abord le lien entre les propriétés d'une opération puis entre opérations et pré-requis.

Les rapports entre les propriétés des opérations

Nous rappelons qu'une opération est définie par ses entrées, sa nature, son déclenchement, son statut, ses sorties, son caractère obligatoire ou facultatif.

Nous examinons les relations entre :

- déclenchement et obligatoire/facultatif
- déclenchement et nature
- nature et obligatoire/facultatif
- déclenchement et événement d'entrée.

Les autres propriétés sont de toute évidence totalement indépendantes.

Rapport entre déclenchement et obligatoire/facultatif

Ces deux propriétés ne sont pas indépendantes. En effet, si une opération est facultative son déclenchement est nécessairement optionnel puisque seul l'utilisateur peut savoir si cette opération doit ou non être déclenchée.

Nous avons l'implication :

facultatif => optionnel

qui équivaut à :

systematique => obligatoire.

Le déclenchement systématique ne peut donc avoir lieu que si l'opération est obligatoire. Par contre, une opération obligatoire peut avoir un déclenchement optionnel ou systématique.

Rapport entre déclenchement et nature

Dans la description de la R.C d'une application interactive, une opération ne peut être qu'interactive ou automatique. Si l'opération est interactive, son déclenchement peut être optionnel ou systématique. Si l'opération est automatique, son déclenchement peut être optionnel ou systématique. Ce qui revient à dire que nature et déclenchement sont indépendants.

Rapport entre nature et obligatoire/facultatif

Ici aussi nous avons deux propriétés indépendantes car une opération interactive ou automatique peut être indifféremment obligatoire ou facultative.

Rapport entre déclenchement et événement d'entrée

Au niveau de la R.C., le déclenchement optionnel peut être considéré comme une forme particulière d'entrée distincte d'un événement du système d'information, en ce sens que l'événement est décrit par une structure de donnée alors que le déclenchement optionnel est décrit par une commande utilisateur. Les événements d'entrée et le déclenchement concourent à la réalisation du pré-requis nécessaire à l'activation de la tâche. Si les événements d'entrée ont eu lieu, cela entraîne l'activation de la tâche si le déclenchement est automatique, sinon cela rend possible le déclenchement optionnel de la tâche. Mais, dans ce dernier cas, la tâche ne sera activée que si l'utilisateur décide d'user de son droit de déclenchement.

Les rapports entre les propriétés des précédences et des opérations

Nous allons maintenant examiner le lien entre les propriétés des opérations et des précédences suivantes :

- celle d'obligatoire ou de facultatif pour les opérations,
- celle de permanent ou indicatif pour les précédences, les autres propriétés étant indépendantes.

La question qui se pose ici est de savoir si certaines combinaisons de ces propriétés ne pourraient pas aboutir à des situations de blocage. Nous examinerons seulement le cas de précedence permanentes car les précédences indicatives ne donnent pas lieu à vérification. Soit $P(O_i, O_j)$ une précédence entre O_i et O_j ; si O_i et O_j sont de même type

(tous les deux obligatoires ou tous les deux facultatifs), il n'y a aucun problème.

Si O_i est obligatoire et O_j facultatif cela signifiera que, une fois O_i exécutée, O_j ne le sera pas forcément car cela dépendra de l'utilisateur. Mais ce cas ne provoquera pas de blocage.

Si O_i est facultatif et O_j obligatoire, O_j ne sera déclenché que si O_i l'est. En effet, si O_i n'est pas exécuté O_j ne pourra pas être déclenché et O_i étant facultatif son déclenchement est nécessairement optionnel c'est-à-dire dépendant de l'utilisateur. Donc si l'utilisateur ne déclenche pas O_i , O_j ne pourra pas être déclenchée car elle ne sera pas activable. Mais ce cas non plus ne sera pas un blocage car la définition d'une opération obligatoire précise que le déclenchement de cette opération est nécessaire pour atteindre le but seulement si elle est activable, ce qui n'est pas forcément le cas.

L'examen de ces cas montre clairement que par le biais des opérations facultatives l'utilisateur peut avoir une marge de manoeuvre importante et que le *comportement de la procédure ne sera pas déterministe*.

Correspondances entre procédure prévue, minimale et effective

Nous avons vu qu'à une procédure prévue correspond dans un contexte donné une procédure minimale et qu'à une procédure minimale correspond plusieurs procédures effectives.

Nous voulons expliciter dans ce paragraphe quelles sont les correspondances entre les différentes tâches afin de pouvoir vérifier la compatibilité entre :

- la procédure prévue et la procédure minimale,
- une procédure effective et la procédure minimale.

Par *compatibilité*, nous voulons dire qu'une *procédure prévue*, une *procédure effective* et une *procédure minimale* décrivent bien la même tâche.

Une procédure prévue pour un poste de travail et un but donnés est décrite par une liste d'opérations et une liste de pré-requis.

La procédure prévue correspond à la procédure standard qui est exécutée dans un cas normal ou pour guider un utilisateur novice.

La procédure prévue pour un poste de travail et un but donnés est décrite par les mêmes éléments que la procédure minimale pour lesquels on peut modifier le pilotage en:

- supprimant des déclenchements optionnels
- rajoutant des précédences indicatives
- augmentant les opérations obligatoires ou les précédences permanentes.

Pour un poste de travail donné, on dira qu'une procédure minimale et une procédure prévue sont compatibles si :

- i) la procédure prévue et la procédure minimale ont le même but
- ii) toutes les opérations obligatoires de procédure minimale sont des opérations de procédure prévue
- iii) toutes les précédences permanentes de procédure minimale sont des précédences de procédure prévue
- iiii) tous les déclenchements systématiques de procédure minimale sont des déclenchements de procédure prévue.

Nous remarquons que l'ensemble des séquences de procédure prévue réalisant le but doit être inclus dans l'ensemble des séquences de procédure minimale réalisant le but. La vérification de cette inclusion est fondamentale car elle permettra à l'analyste de vérifier que la procédure minimale qu'il vient de définir lui permettra d'exécuter la procédure prévue correspondante.

Une fois la procédure minimale définie, il nous faudra vérifier la compatibilité de toutes les procédures effectives que nous créerons par rapport à la procédure minimale. La procédure prévue et les procédures effectives ont la même définition de compatibilité par rapport à procédure minimale car la procédure prévue peut être considérée par rapport aux validations comme une procédure effective particulière.

Mais la distinction fondamentale entre une procédure prévue et une procédure effective est d'ordre *méthodologique*.

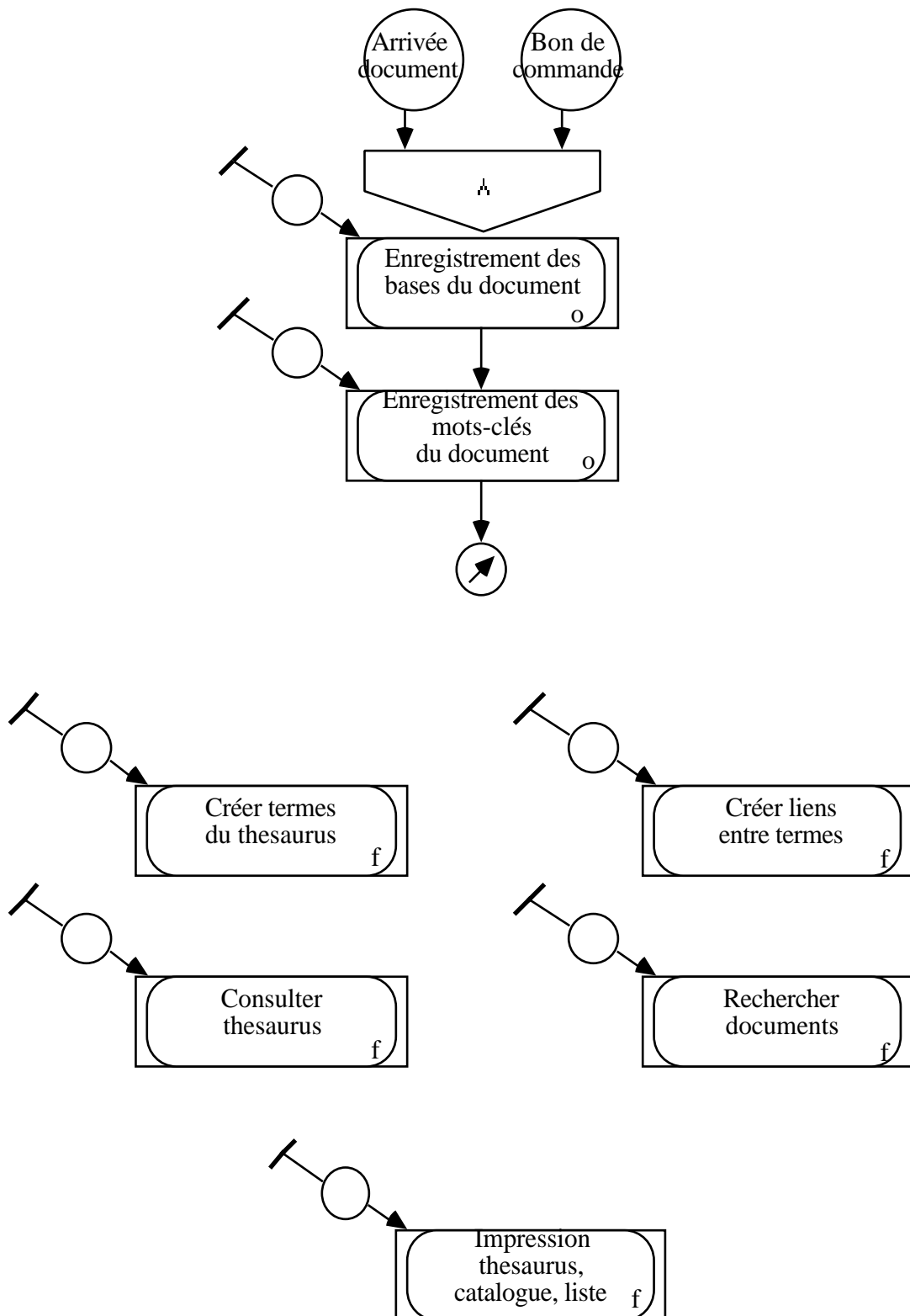
La procédure prévue correspondant généralement à la première observation de la tâche que nous pouvons faire et qui est, par définition, la tâche standard alors que les procédures effectives correspondent à toutes les variantes apportées par les utilisateurs en situation réelle de travail et qui seront définies au fur et à mesure des besoins.

2.2.2.4 Formalisme graphique

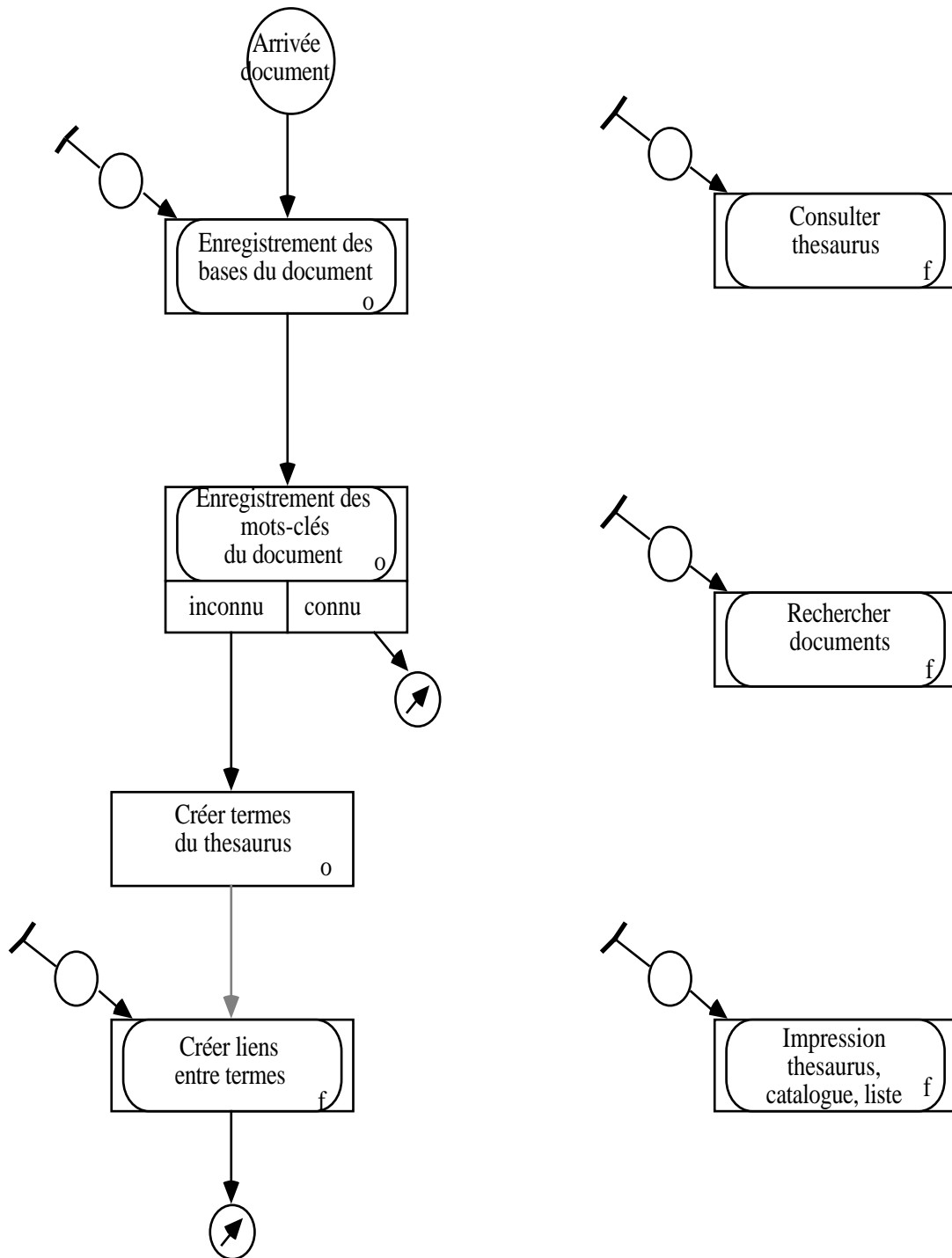
Nous définissons un formalisme graphique qui permet de décrire l'ensemble des concepts que nous avons définis dans la R.C.

Concept	Représentation graphique
Opération automatique	
Opération interactive	
Opération manuelle	
Opération obligatoire	
Opération facultative	
Règle d'émission	
Evènement (entrée ou sortie d'opération)	
Evènement terminal	
Déclenchement optionnel	
Déclenchement automatique	Par défaut (si pas optionnel)
Pré-requis	
Synchronisation	Equation booléenne sur les évènements d'entrée (figure dans le symbole du pré-requis) . Ex :
Précédence permanente	
Précédence indicative	

Exemple : procédure minimale d'uposte de documentation



Exemple : une procédure effective d'un poste de documentation



Commentaires sur l'exemple

L'exemple ci-dessus est extrait de la Représentation Conceptuelle d'un poste de documentation. Il concerne le but "Enregistrement de nouveaux documents".

Pour satisfaire ce but, le documentaliste a besoin de disposer des opérations de mise à jour du thesaurus et de consultations diverses; c'est pourquoi ces opérations figurent dans la description de la procédure minimale.

Toutes les précédences permanentes et les opérations obligatoires de la procédure minimale doivent figurer dans toutes les procédures effectives; c'est le cas, ici, de la précedence permanente entre les opérations obligatoires "Enregistrement des bases du document" et "Enregistrement mots-clés du document".

Il est possible, par contre, de rajouter des précédences permanentes et des opérations obligatoires dans une procédure effective pour minimiser les manipulations de l'utilisateur dans les cas usuels; c'est le cas, ici, pour la précedence entre les opérations "Enregistrement mots-clés du document" et "Créer termes du thesaurus" qui devient permanente dans la procédure effective ainsi que l'opération "Créer termes du thesaurus" qui devient obligatoire et automatique.

Mais l'opération "Créer liens entre termes" est restée interactive, facultative et à déclenchement optionnel car c'est une opération complexe pour le documentaliste et il n'est pas évident qu'il ait envi de l'effectuer ponctuellement pour tout nouveau mot-clé enregistré dans le thesaurus.

Le même formalisme s'applique à tous les différents niveaux de détails; chaque opération est reprise et détaillée jusqu'au niveau de l'opération élémentaire dans la communication homme-machine, c'est-à-dire une entrée ou une sortie d'une transaction.

Un exemple complet est montré dans le chapitre 5.

Il est important de noter que seuls les paramètres variables sont exprimés sur ce schéma; pour des raisons de lisibilité les paramètres constants que nous développons au paragraphe suivant sont volontairement omis.

En conséquence, ces schémas doivent être interprétés comme la répartition du pilotage entre l'homme et l'ordinateur en ce qui concerne les paramètres variables auxquels viennent s'ajouter de manière implicite toutes les possibilités des paramètres constants.

2.2.3 Les paramètres constants de la Représentation Conceptuelle

Ces paramètres vont constituer des aides à l'utilisateur utilisables quelle que soit la tâche effectuée par lce dernier. Le caractère

général de ces paramètres fait qu'ils n'ont pas besoin d'être spécifiés pour chaque application.

Nous distinguerons les aides à la réalisation de l'activité, les aides à l'apprentissage et les possibilités d'évolution du logiciel.

2.2.3.1 Aides au travail

Interruption

L'interruption permet à l'utilisateur de quitter n'importe quelle opération en cours d'exécution pour aller exécuter une autre opération et reprendre la première opération à son point d'interruption.

On peut prévoir plusieurs niveaux successifs d'interruption, bien que concrètement, au-delà de trois ou quatre niveaux, ce soit difficilement gérable par l'utilisateur.

Cette notion est particulièrement importante pour toutes les opérations de consultation auxquelles on veut accéder alors qu'une autre opération est active. Notre hypothèse est qu'en système monoposte, toutes les opérations sont possibles à tout moment. Pour les systèmes en temps partagé, cela nécessite que le système gère les conflits éventuels concernant les mises à jour multiples de fichiers.

Transfert de données

La possibilité est donnée à l'utilisateur de transférer une donnée d'une opération à une autre sans avoir à la ressaisir. Cette fonction permet de minimiser les saisies et donc d'éviter les erreurs de recopie; elle permet également, lors des interruptions, de transporter une donnée dans plusieurs opérations successives.

Quitter

L'utilisateur doit pouvoir demander l'abandon de son travail à n'importe quel moment sans avoir à finir une opération ou à repasser par une arborescence du menu. Le logiciel peut lui signaler les conséquences de cette demande selon l'endroit où elle se situe.

Différer

L'utilisateur peut différer n'importe quelle opération dans le temps afin de la reprendre ultérieurement à sa convenance. Différer se distingue de quitter en ce sens que dans l'action de différer il y a mémorisation de l'opération non terminée ce qui n'est pas le cas dans l'action de quitter.

Annuler

L'utilisateur peut annuler la dernière action qu'il vient de faire; cette possibilité est particulièrement utile pour le redressement des erreurs perçues par l'utilisateur ou signalées par le logiciel.

Mémorisation de l'activité

Ce modèle d'interface n'imposant pas de cadre rigide à l'opérateur, celui-ci peut connaître des difficultés de mémorisation de ce qu'il a accompli.

Plusieurs types de mémorisation lui seront donc proposés:

- entre les sessions, l'enregistrement, pour un utilisateur, des opérations qu'il a différées dans le temps. Cet enregistrement lui permettra à tout moment de connaître l'ensemble des opérations qui sont "en cours" et qu'il devra finir un jour.

- entre les sessions, la mémorisation des paramètres propres à l'utilisateur: identification, vocabulaire propre, procédures effectives...

- en cours de session, l'enregistrement des interruptions successives et non terminées qui sont en train d'être réalisées par l'utilisateur. Ceci lui permet surtout de ne pas se perdre dans le logiciel, en particulier s'il est lui-même interrompu par un événement extérieur.

- en cours de session, on peut avoir aussi une trace du travail total ou à défaut des deux ou trois dernières opérations réalisées afin de pouvoir remonter facilement à l'état précédent du système. Cette trace de l'activité peut être intéressante pour la détection et le redressement des erreurs constatées par l'utilisateur (en l'absence de possibilité de retour-arrière automatique, ce qui est difficile à réaliser en toute généralité).

2.2.3.2 Aides à l'apprentissage

Nous voulons parler ici d'aide à l'apprentissage du maniement du logiciel proposé et non pas du travail lui-même.

Nous proposons deux types d'aide qui sont le guidage fonctionnel et le guidage d'utilisation.

Guidage fonctionnel

Une commande de type SOS doit permettre à tout moment à l'utilisateur de connaître soit la liste des opérations possibles dans

l'état actuel d'avancement de son travail, soit une explication des fonctions et effets d'une commande donnée.

Ce type de possibilité est maintenant classique sur un grand nombre de logiciels.

Guidage d'utilisation

A tout moment, l'utilisateur doit pouvoir passer à un mode guidé selon une logique d'utilisation où un enchaînement d'opérations correspondant aux précédences permanentes et indicatives lui sera proposé en fonction du but qu'il s'est fixé.

Ce type de guidage peut permettre de répondre à des questions du type "Comment faire pour? ". Ceci est rendu possible grâce à la structuration du système en buts, sous-but, opérations et précédences.

2.2.3.3 Les possibilités d'évolution

Nous parlons ici essentiellement d'évolution hors programmation, c'est-à-dire qui peuvent être mises en œuvre, éventuellement, par l'utilisateur lui même.

Ce qu'il sera aisé de modifier concerne:

- la création de nouvelles procédures effectives adaptées aux modes de travail des différents utilisateurs
- la répartition du pilotage entre l'homme et l'ordinateur c'est-à-dire les notions de déclenchement, d'opérations obligatoires ou facultatives, de précédences permanentes ou indicatives
- les niveaux d'opérations intermédiaires qui n'ont pas été initialement prévus mais qui utilisent des opérations déjà connues de la machine. Cela peut permettre à l'utilisateur de se créer ses propres séquences standard ou des macro-opérations originales

Si on veut faire appel à de nouveaux traitements qui ne peuvent pas être composés à partir des opérations déjà connues de la machine, il sera nécessaire de programmer ces nouveaux modules. Il en va de même si l'on veut faire apparaître des données ou des relations entre données non décrites dans les structures de données de la machine.

Cependant, la représentation interne que nous proposons au chapitre cinq pour implémenter ce système est orientée vers une facilité maximum d'évolution.

Les paramètres constants des applications interactives que nous venons de décrire doivent être présents constamment dans l'esprit de l'analyste pour deux raisons:

- ils facilitent et allègent la description de la représentation conceptuelle de chaque application interactive ; en effet, l'analyste

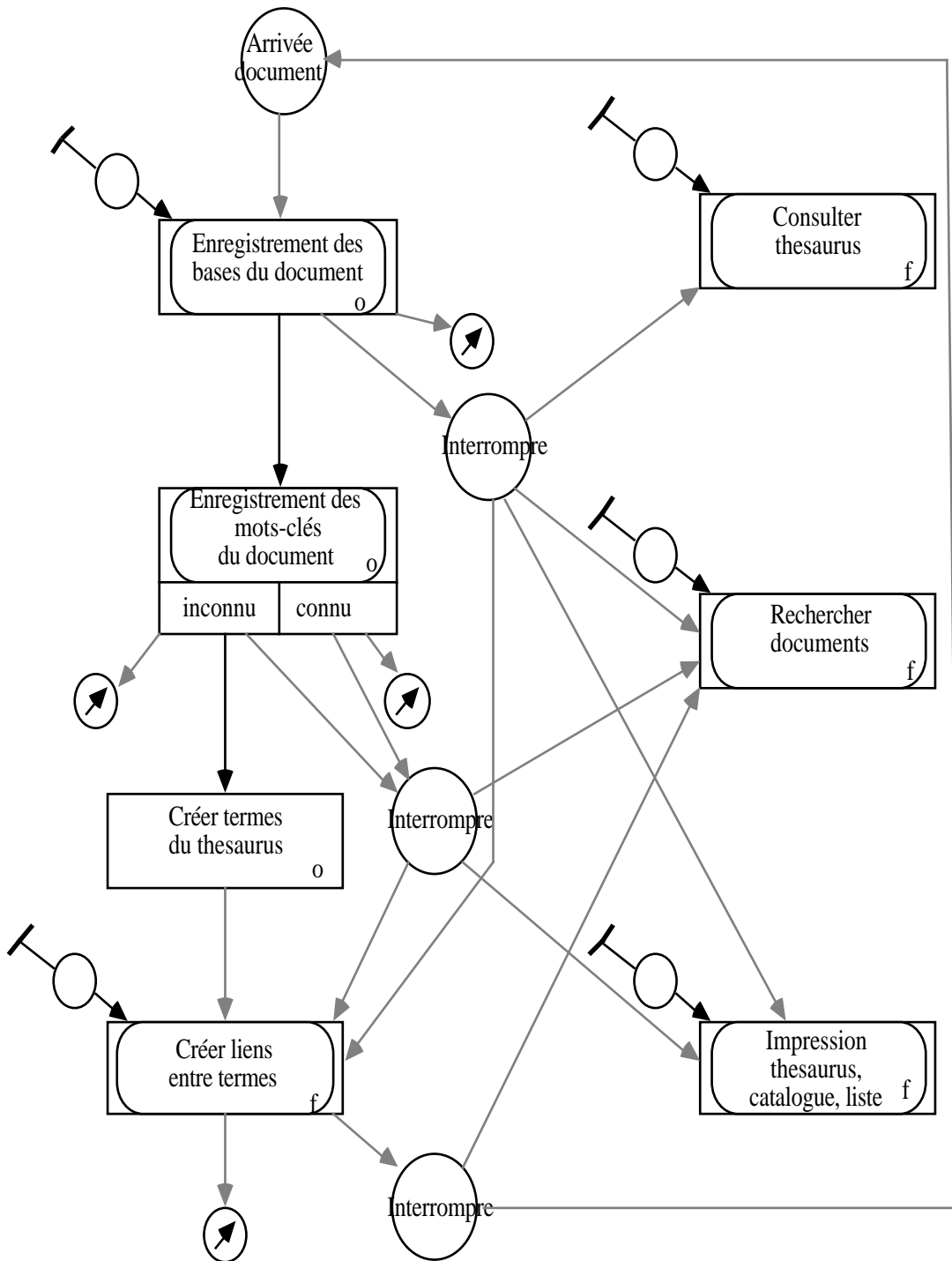
n'a plus besoin de spécifier les points de retour possibles au menu principal ou aux opérations de consultation puisqu'il est possible à tout moment de quitter, de différer ou d'interrompre;
- inversement, ils permettent d'interpréter différemment les schémas de la représentation conceptuelle.

Pour se convaincre de la nécessité de ne pas faire figurer les paramètres constants, nous avons visualisé sur le schéma suivant qui reprend l'exemple de la procédure effective de la documentaliste ce qui se passerait si on indiquait par des flèches toutes les possibilités d'interrompre et de quitter de l'utilisateur.

Nous voyons clairement que les schémas de la Représentation Conceptuelle deviendraient alors illisibles.

En pratique, il n'y a que dans le cas où les possibilités standard de l'utilisateur sont réduites qu'on peut les faire figurer dans les schémas.

Contre-exemple : visualisation de commandes Quitter et Interrompre



Chapitre 3 :

LES ASPECTS DE SURFACE DES APPLICATIONS INTERACTIVES

Ce chapitre a pour but de présenter les principes de conception de la Représentation Externe qui correspond à ce que voit l'utilisateur de l'application interactive.

Comme nous le rappelons dans le schéma ci-dessous, la Représentation Externe provient d'une part des concepts de la Représentation Conceptuelle, d'autre part de critères de psychologie cognitive ou d'ergonomie.

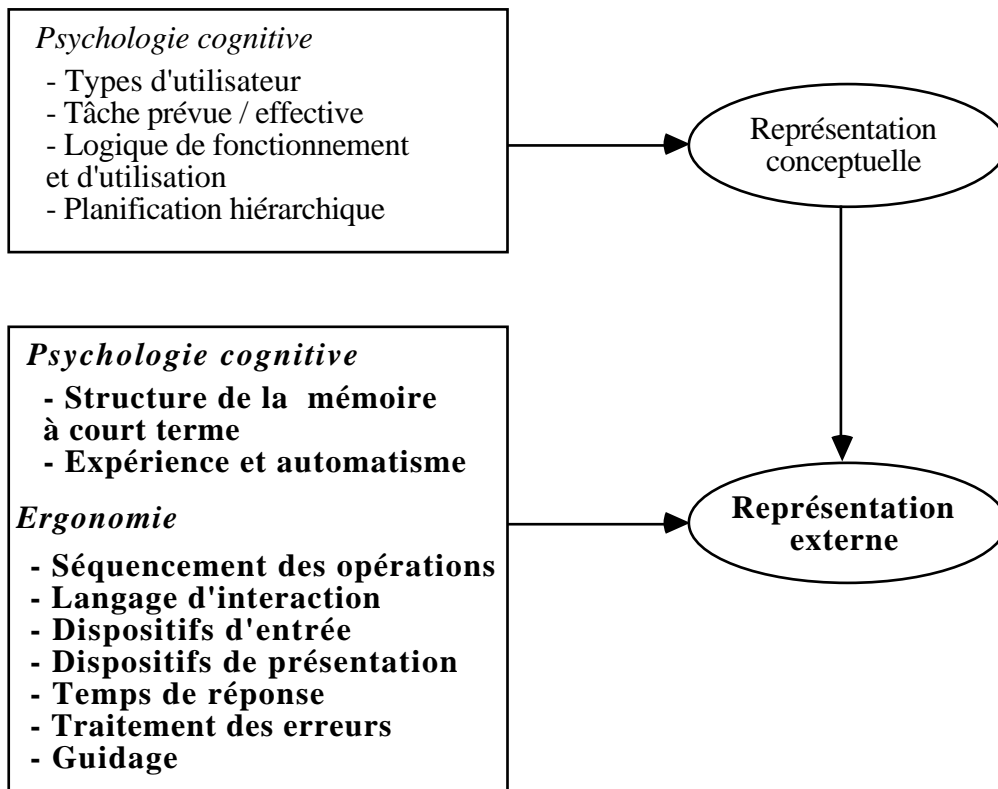
Il faut aussi noter que la conception de la Représentation Externe est fortement dépendante de contraintes techniques qui sont d'une part les caractéristiques physiques de l'écran, d'autre part les logiciels de gestion d'écran. Pour mettre en valeur ces contraintes, nous donnons des exemples de mise en oeuvre avec un logiciel de gestion d'écran classique et avec un logiciel de multifenêtrage.

Ce chapitre est divisé en trois parties.

Dans la première partie, nous présentons les connaissances de psychologie cognitive sur la structure de la mémoire et sur l'apprentissage car elles permettent de mieux comprendre le fondement des recommandations ergonomiques.

Dans la deuxième partie, nous exposons les recommandations ergonomiques concernant les sept paramètres de l'interface et nous citons des expériences d'observations directes.

Dans la troisième partie, nous montrons comment intégrer ces éléments à la conception de la Représentation Externe compte-tenu des contraintes des logiciels de gestion d'écran.

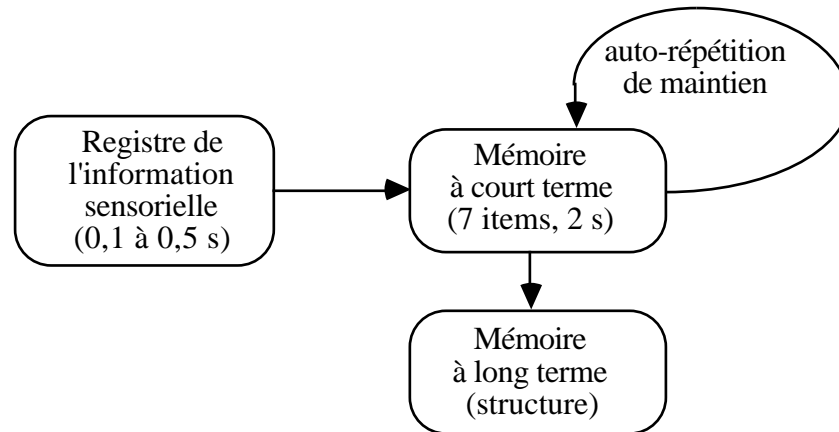


3.1 L'apport de la psychologie cognitive à la Représentation Externe

3.1.1 Les structures de la mémoire humaine

La perception et l'enregistrement d'une information quelconque chez l'être humain met en cause trois types de mémoire:

- un *registre de l'information sensorielle* qui est capable de contenir l'intégralité d'une image pendant un temps très court (de 0,1 à 0,5 secondes),
- une *mémoire à court terme* qui interprète les informations contenues dans le registre de l'information sensorielle afin d'en extraire les éléments significatifs,
- une *mémoire à long terme* que l'on peut considérer comme sans limite par rapport à la longueur d'une vie humaine.



Nous ne développons pas ici les caractéristiques du registre de l'information sensorielle car elles relèvent de la perception physiologique et concernent par conséquent l'ergonomie du matériel. Les mémoires à court terme et à long terme ont par contre un impact important sur l'ergonomie du logiciel.

La mémoire à court terme constitue une limite énorme pour l'être humain; en effet et schématiquement, elle ne peut retenir que 7 *items* pendant un délai de deux secondes.. Un *item* correspond à une unité d'information significative pour un individu.

Exemple d'items:

H C T correspond à trois items

CHAT correspond à un item (pour une personne connaissant le français)

Le chiffre de 7 items correspond à une moyenne qui fluctue entre un minimum de 5 et un maximum de 9 selon les personnes.

Le délai de deux secondes de rétention de l'information sans altération et sans effort est un minimum atteint par tout le monde. Au-delà, cette information s'altère avec le temps; au bout de trois secondes, il n'y a plus que 80% de réponses justes et à six secondes 40%. Si l'on désire conserver une information plus longtemps dans la mémoire à court terme, il est nécessaire de procéder à une autorépétition de maintien qui consiste à répéter mentalement cette information jusqu'à son utilisation.

C'est ce que nous pratiquons spontanément quand nous voulons nous souvenir d'un numéro de téléphone que nous ne pouvons écrire immédiatement.

L'effacement de la mémoire à court terme se produit spontanément avec le temps, mais il peut être provoqué aussi par interférence avec d'autres informations; en effet, l'arrivée de nouvelles informations que nous désirons mémoriser effacera une partie ou la totalité des

informations présentes dans la mémoire à court terme et ces dernières seront définitivement effacées si nous n'avons pas eu le temps de les faire basculer de la mémoire à court terme vers la mémoire à long terme ou de les transcrire sur papier.

Les mécanismes de mémorisation et de recherche dans la mémoire à long terme ne sont pas connus avec autant de précision que ceux de la mémoire à court terme.

Cependant, nous savons que l'effort de mémorisation dans la mémoire à long terme requiert une *concentration* qui nous rend relativement indisponible pour d'autres tâches. Par exemple, nous savons qu'il est difficile pour un étudiant d'écrire complètement une phrase entendue et de la mémoriser en même temps; cette difficulté est accrue s'il ne comprend pas cette phrase. Ce dernier phénomène est dû, d'une part aux limites de la mémoire à court terme, d'autre part au fait que la mémorisation dans la mémoire à long terme est facilitée par la compréhension et par la possibilité de *rattacher toute nouvelle information à une information déjà existante dans nos structures mentales*. En d'autres termes, il est plus facile de retenir une information dans un domaine connu et qui complète des informations que nous possédons déjà ou dit autrement "plus on sait, plus on est capable d'apprendre".

Ce fait explique l'intérêt du raisonnement par analogie qui permet de comparer un phénomène inconnu à un phénomène connu.

La notion de compréhension est liée à la notion de structure de l'information; cela suppose que l'information que l'on veut mémoriser a une structure significative et que cette structure a une correspondance avec la structure des informations que nous possédons déjà. Une conséquence immédiate est qu'*une information structurée est plus facilement mémorisable*.

A partir de ces faits, différents modèles explicatifs du fonctionnement de la mémoire ont été élaborés, mais nous ne les exposerons pas ici car il ne s'agit que d'hypothèses et ils n'ont pas de conséquences pratiques en ergonomie du logiciel.

Les conséquences de cette structure de la mémoire sur les interfaces sont multiples:

- une simple interruption (téléphone, client,...) crée un effacement de la mémoire à court terme; pour remédier à cette limite, il est nécessaire de mémoriser l'activité de l'utilisateur de manière à pouvoir lui rappeler ces dernières actions, de lui signaler ces erreurs le plus tôt possible, de lui signaler les conséquences de ces actions,

- la dégradation rapide de l'information de la mémoire à court terme avec le temps doit être prise en compte dans la conception du paramètre "temps de réponse",
- les limites de capacité en quantité d'information de la mémoire à court terme nécessitent la concision et l'emploi de concepts les plus agrégés possibles sous la forme la plus identifiable correspondant ainsi (après apprentissage) à un seul item,
- la mémorisation en mémoire à long terme est facilitée par la compatibilité entre les anciennes et les nouvelles procédures ou par des analogies avec des systèmes connus.

3.1.2 Expérience et automatisme

Nous avons déjà parlé dans le chapitre précédent de la différence entre utilisateurs novices et utilisateurs expérimentés en insistant sur le fait qu'ils n'ont pas la même représentation de leur travail, nous voulons présenter ici une autre différence qui a un impact important sur les interfaces.

L'utilisateur qui répète souvent les mêmes tâches sur des périodes suffisamment longues acquiert des *automatismes* qui lui permettent d'exécuter ces tâches très rapidement et avec une mobilisation minimale de ces processus conscients.

En effet, l'exécution de ces tâches devient en partie ou totalement inconsciente et l'efficacité du travail s'en trouve notablement accrue. Par analogie avec l'ordinateur, on pourrait dire que ces tâches sont devenues "cablées".

Nous pouvons citer comme exemple l'apprentissage du clavier ou la conduite de sa voiture sur un itinéraire quotidien.

De manière schématique, nous pouvons même dire qu'un des objectifs essentiels de tout apprentissage est de créer chez l'individu le maximum d'automatismes pour toutes les tâches répétitives afin de lui permettre de dégager le maximum de temps pour des activités plus rares et plus complexes.

Une conséquence immédiate de cette connaissance est que la conception de l'interface doit favoriser la création de ces automatismes. Pour cela nous devons concevoir tous les paramètres avec le maximum d'*homogénéité*.

En effet, la création d'automatismes ne peut se produire que si dans toutes les situations identiques l'action de l'utilisateur est la même.

Par exemple, la touche permettant de valider l'entrée des données doit être la même pour toutes les applications informatiques.

Ce souci d'homogénéité a un impact sur presque tous les paramètres de l'interface:

- en entrée: touches/fonction ou souris
- en sortie: disposition spatiale de l'écran
- erreurs: présentation et rectification
- langage: syntaxe et vocabulaire
- séquençement: procédure d'enchaînement
- guidage: présentation et concepts

3.2 L'apport de l'ergonomie à la Représentation Externe

3.2.1 Les paramètres de l'interface

Comme nous l'avons vu au chapitre 1, l'interface vue par les ergonomes se décompose en sept paramètres qui sont:

- le séquençement des opérations
- le langage d'interaction
- les dispositifs d'entrée
- les dispositifs de sortie
- le temps de réponse
- le traitement des erreurs
- le guidage

Mais ces paramètres ne sont pas homogènes et, en particulier, ils ont des impacts différents sur l'utilisateur.

Pour illustrer ce fait, nous présentons une modélisation de l'opérateur en trois niveaux faite par Falzon (Fal, 82) à propos du contrôle aérien:

- Le premier niveau concerne les caractéristiques physiologiques de l'opérateur; il fait référence aux travaux concernant la perception physiologique comme, par exemple, les limites de lisibilité sur un écran, la fatigue visuelle, etc...

- Le deuxième niveau concerne le traitement par l'utilisateur des opérations élémentaires. La mémorisation du vocabulaire ou l'utilisation correcte d'une syntaxe fait partie de ce niveau .

Ce niveau regroupe les aspects lexicaux et syntaxiques utilisés en linguistique.

- Le troisième niveau concerne le système de représentation et de traitement de l'information de l'utilisateur qui lui permet de synthétiser, résoudre un problème en fonction des informations disponibles. Par exemple, la représentation d'une procédure de

traitement ou la résolution d'un problème fait partie de ce dernier niveau.

Ce niveau regroupe les aspects sémantiques et contextuels de la linguistique.

Si on considère les différents paramètres de l'interface, on s'aperçoit qu'ils n'ont pas le même statut vis-à-vis des trois niveaux de l'utilisateur. Par exemple, le séquençement des opérations fait appel aux niveaux II et III de l'utilisateur.

Nous pouvons représenter l'ensemble de ces effets sur le tableau suivant :

Niveaux Paramètres	I. Physiologique	II. Opérations élémentaires	III. Système de représentation et de traitement
Séquencement des opérations mode opératoire		×	×
Langage d'interaction		×	×
Dispositifs d'entrée	×		
Dispositifs de sortie	×	×	
Temps de réponse			×
Traitement des erreurs		×	×

Dans la suite de ce travail, nous ne nous intéressons pas aux effets physiologiques qui relèvent de l'ergonomie du matériel mais seulement aux niveaux II et III qui relèvent de l'ergonomie du logiciel. Aussi, nous n'exposerons pas ici les effets physiologiques des dispositifs d'entrée et de sortie bien qu'ils aient fait l'objet de très nombreux travaux en ergonomie.

3.2.2 Recommandations ergonomiques pour les paramètres de l'interface

Nous présentons des recommandations ergonomiques sur les sept paramètres énumérés précédemment.

Le sujet est trop récent pour que ces recommandations soient exhaustives et en outre la combinaison de ces recommandations peut aboutir à des contradictions partielles qui ne peuvent être levées que par une expérimentation auprès des utilisateurs.

3.2.2.1 Séquencement des opérations

Le problème évoqué ici est celui de l'adéquation entre l'ordre des opérations fixé par la machine et celui nécessaire à l'opérateur pour effectuer sa tâche dans n'importe quel environnement.

En effet, la plupart du temps la conception de l'interface se fait en fonction du déroulement normal (tâche prévu) ou moyen de la tâche sans tenir compte des types d'utilisateurs (expérimenté / novice) ou d'aléas provenant de l'environnement; alors que, ces éléments constituent le contexte naturel de toute tâche effectuée par un être humain .

Nous rapellons que la plupart des caractéristiques du séquencement font partie de la structure profonde des logiciels et sont à ce stade déjà déterminées, il s'agit :

- du type d'enchaînement (libre, guidé, automatique)
- des possibilités d'enchaînements variés selon le degré d'expérience et le type d'utilisateur
- d'accéder à toutes les informations avec autant ou plus de facilité qu'à la main
- de pouvoir quitter ou annuler à tout moment
- d'interrompre à tout moment une transaction en cours soit pour un temps bref (consultation), soit pour la reprendre ultérieurement (différer)
- d'afficher ce que l'on vient de faire afin de reprendre plus facilement après une interruption
- de faire appel à une opération quelconque à partir d'une autre et de revenir en arrière
- de transférer des informations d'une opération à l'autre.

La partie externe du logiciel devra permettre à l'utilisateur d'exécuter le plus facilement possible l'ensemble des possibilités

ainsi offertes c'est-à-dire qu'au niveau de l'interface, il faudra traduire ces caractéristiques en:

- dispositifs de sortie et d'entrée
- vocabulaire et syntaxe
- messages d'erreur et guidage,

ainsi que nous allons le voir dans les paragraphes suivants.

3.2.2.2 Langage d'interaction

Le langage d'interaction est l'outil qui va permettre à l'utilisateur d'exprimer, au moyen d'un vocabulaire et d'une syntaxe, les opérations qu'il désire faire effectuer à la machine. Ces opérations portent sur des commandes et / ou des données.

Le vocabulaire

De façon unanime, les ergonomes privilégient l'emploi du langage des spécialistes de la tâche plutôt qu'un langage d'informaticien. Or, à l'heure actuelle, on ne peut pas dire que l'on trouve ce type de langage dans tous les logiciels développés.

A celà, plusieurs raisons :

- la première, de fond, est liée à la définition même du langage d'interaction qui est un compromis entre les tâches de l'utilisateur et les fonctions informatiques
- la seconde, plus circonstancielle, est liée au fait que ces langages sont conçus par des informaticiens qui pensent en fonction de la machine et de leur propre formation (voir l'étude faite par Scapin (Sca,81) d'un langage de messagerie électronique)
- la dernière est liée à la création de tâches nouvelles ou traitées différemment à cause de l'automatisation (traitement de texte, messagerie électronique) et qui nécessitent donc très souvent la création d'un vocabulaire nouveau.

Dans le cas où l'on ne peut utiliser des mots du vocabulaire courant ou du vocabulaire professionnel en leur conservant le même sens, que peut-on choisir comme vocabulaire du langage de commande :

- des codes numériques ?
- des codes mnémoniques ?
- des mots de la langue française ?
- des mots inconnus ?
- des pictogrammes?

Les codes numériques font l'unanimité contre eux par leurs difficultés à être mémorisés et les risques d'erreurs élevés. Leurs seuls avantages résident dans le fait qu'ils sont courts et donc rapides à saisir.

Cependant les codes numériques qui peuvent être décomposés selon une structure connue sont plus faciles à mémoriser (comme nous l'avons vu pour la mémorisation à long terme).

Les codes mnémoniques sont préconisés assez fréquemment mais sans qu'il y ait accord sur leurs règles de fabrication : faut-il prendre la première lettre du mot, les deux premières lettres, les trois premières lettres ou d'autres ? Nous manquons ici d'expériences sérieuses nous disant quelles règles de tronçatures assureraient une meilleure mémorisation et un moindre risque d'erreur. Peut-être n'existe-t-il pas de règles générales et faut-il tester ceci pour chaque langage ? Certains préconisent de laisser aux utilisateurs le soin de créer eux-mêmes leurs propres abréviations à partir des mots complets.

Les mots du langage naturel : de nombreux informaticiens estiment que l'utilisation du langage naturel est ce que l'on peut offrir de mieux à l'utilisateur. C'est pourquoi l'évaluation d'un langage (Sca,81) développé dans le cadre d'une messagerie électronique est intéressante car le résultat est absolument contre-intuitif. En voici les principales conclusions :

- le vocabulaire est mieux compris par les informaticiens que par les utilisateurs naïfs ; ce résultat provient du fait que ce vocabulaire ayant été choisi par les informaticiens (inconsciemment en fonction de leur culture), les mots utilisés avaient un sens différent dans le contexte informatique par rapport au langage courant
- les termes choisis ne sont pas évidents sans apprentissage
- il y a de nombreuses confusions dues à des problèmes de synonymie et d'indétermination syntaxique
- la mémorisation est très mauvaise car il semble, à la limite, plus difficile de mémoriser un mot connu avec un nouveau sens que de mémoriser un mot nouveau

Les mots inconnus (créés ou issus d'une langue étrangère) : il n'y a plus de risque de confusion comme précédemment, mais il y a un effort de mémorisation plus important.

Les pictogrammes peuvent se révéler très intéressants quand ils sont sans ambiguïté pour l'utilisateur; leur intérêt réside dans le fait qu'ils permettent d'exprimer un concept sous un seul item facilement mémorisable dans la mémoire à court terme, et qu'ils peuvent permettre l'analogie avec un cas manuel connu ce qui facilite la mémorisation à long terme.

De tout cela, se dégage le fait qu'il n'y a pas une solution qui l'emporte nettement; il y a donc lieu de procéder par expérimentation dans chaque cas. D'où l'intérêt d'un prototype d'interface sur lequel on peut expérimenter et que l'on peut modifier aisément.

La syntaxe

Ce sont les règles qui permettent d'exprimer les commandes ou les données et de les combiner entre elles.

Les diverses études se rejoignent sur la nécessité de deux caractéristiques:

- *simplicité* pour des raisons de mémorisation
- *homogénéité* pour éviter les risques d'erreurs. En effet si la syntaxe est homogène, l'utilisateur peut développer une automatisation de son comportement en tapant, par exemple "retour chariot" après chaque commande. Si cette règle a des exceptions, ces exceptions seront fatalement des sources d'erreurs. Il faut noter que si l'homogénéisation est facile à atteindre au niveau d'une application, elle l'est plus difficilement d'une application à l'autre et elle nécessite alors une conception globale de la syntaxe du langage d'interaction qui intègre toutes les applications.

La conception d'un langage d'interaction ne se résume pas à ces deux critères. L'objectif d'un langage d'interaction est d'être d'un apprentissage et d'une utilisation faciles pour l'utilisateur.

Pour illustrer ces deux notions, nous nous référons à une étude faite par Corson (Cor,82) et portant sur la comparaison de deux langages d'interrogation de base de données SQL et QBE.

SQL est un langage d'interrogation classique par mots clés et QBE est un langage dit "graphique" où la syntaxe est allégée au maximum.

Nous donnons un exemple de chacun de ces langages qui traduit la question suivante:

"Trouver le nom des employés qui travaillent dans le département 50 ?"

s'écrit en SQL SELECT NOM

FROM EMP
WHERE DEPT NO = 50

s'écrit en QBE

EMP	NAME	DEPTNO	SAL
	P.DUPONT	50	

où seule la deuxième ligne est remplie par l'utilisateur.

Les études évaluatives de SQL montrent que, tant en ce qui concerne l'apprentissage que l'utilisation effective du langage, les programmeurs réalisent de meilleures performances que les non-programmeurs.

Les études comparatives entre SQL et QBE dans les mêmes conditions d'apprentissage de procédures et de test montrent que QBE:

- requiert moins de temps d'apprentissage
- nécessite moins de temps pour les tests
- permet une plus grande exactitude dans l'établissement des requêtes
- provoque un meilleur niveau de confiance.

On peut supposer que la supériorité ergonomique de QBE est essentiellement due à la structure tabulaire préexistante sur l'écran et qui fournit à l'utilisateur une image de la structure de la base de données lui évitant ainsi la mémorisation de la structure de cette base. On peut remarquer ainsi que QBE n'utilise que très peu de mots-clés, ce qui permet d'éviter des confusions avec le langage naturel, comme pour les ET/OU de SQL.

Pour conclure cette étude de l'interrogation des bases de données, nous devons noter que les difficultés d'apprentissage et d'utilisation ne sont pas dues qu'à des problèmes de présentation et de langage d'interaction. Les difficultés peuvent être liées à la compréhension de la question en langage naturel ou au problème de la transposition par rapport à une structure de données prédéfinie. On remarque aussi un certain nombre d'erreurs liées à l'utilisation du ET/OU logique qui n'a pas la même signification que le et/ou du langage naturel et aussi une difficulté à employer des quantificateurs (25% des erreurs des utilisateurs dans QBE).

Ainsi, sans réduire le problème de l'interrogation des bases de données à celle du choix du langage d'interaction et de présentation, nous voyons que celui-ci a une importance non négligeable sur la facilité d'apprentissage et la minimisation des erreurs.

3.2.2.3 Les dispositifs d'entrée

Un des rôles essentiels des dispositifs d'entrée à part la saisie proprement dite est de permettre la distinction entre commandes et données du point de vue de l'informaticien.

La saisie des données en informatique des organisations ne pose pas de problèmes particuliers car elle se fait obligatoirement par l'intermédiaire des touches alphanumériques du clavier. L'entrée vocale et la désignation directe sur écran sont fort peu utilisées et utiles.

La saisie des commandes offre une plus grande variété:

- frappe au clavier du mot entier ou abrégé désignant la commande
- frappe de touches spécialisées ou touches fonctions
- frappe d'une association de touches, en général une touche spécialisée en même temps qu'une touche banalisée
- désignation par positionnement du curseur et validation, soit par les touches directionnelles du clavier, soit par la souris
- désignation directe sur l'écran par l'intermédiaire d'un light pen ou du doigt

Ces différents modes de saisie ne sont pas équivalents du point de vue de l'utilisateur. Les deux derniers cités sont les moins coûteux sur le plan de la mémorisation; ce peut être le cas aussi de la saisie des premiers caractères de la commande si celle-ci est visualisée sur l'écran. Ces modes de saisie sont donc à préconiser dans le cas d'utilisateurs novices ou intermittents.

Par contre les touches spécialisées peuvent s'avérer très efficace pour des personnes utilisant intensivement le logiciel et habituées à l'usage du clavier.

L'association de touches, très utilisée par les informaticiens, est le dispositif d'entrée le plus complexe pour l'utilisateur; il est difficile à mémoriser et à manipuler et entraîne ainsi de fréquentes erreurs.

3.2.2.4 Les dispositifs de présentation

Le travail interactif avec l'ordinateur se déroulant principalement devant un écran, il nous a semblé important de savoir si l'utilisateur avait des stratégies d'exploration d'écran qui nous permettraient de concevoir des présentations adaptées.

Nous nous sommes rendus compte qu'il faut distinguer deux catégories de cas, selon que l'utilisateur explore l'ensemble de l'écran pour découvrir ce qu'il cherche ou qu'il recherche de manière sélective une information qu'il pense être à une certaine place. Nous parlerons dans le premier cas de "recherche systématique" et dans le second cas de "recherche sélective".

Recherche systématique

Pour ce type d'exploration d'écran, nous nous réfèrons à une expérience faite par Sperandio et Gonju (Spe,83) qui porte sur l'exploration visuelle de tableaux numériques où l'opérateur cherche un nombre donné sans connaître sa localisation.

La première constatation que l'on peut faire est que, dans les pays occidentaux, la lecture d'un texte apparaissant sur un support quelconque s'effectue a priori dans le sens gauche/droite et haut/bas.

Dans le cas des tableaux, l'expérience montre que la stratégie d'exploration est au départ une stratégie par ligne (gauche/droite) et évolue ensuite par colonne (haut/bas) l'inverse. On note également une diminution progressive des durées de fixation et, corrélativement, une diminution du nombre de fixations. La stratégie d'exploration par colonne est adoptée car elle est la plus économique en ce sens qu'elle permet plus facilement une fixation sur le premier chiffre du nombre.

Cette expérience tend à montrer que ce que l'utilisateur voit le mieux est ce qui est situé dans le coin supérieur gauche ou dans la zone centrale. Par contre les nombres situés en extrémité de ligne ou de colonne ont une moins grande probabilité d'être perçus avec précision, d'autant plus que les écrans bombés distordent plus ou moins l'image sur les bords.

Avant de conclure sur la recherche systématique, nous voudrions relativiser les résultats précédents par le fait que les expériences portent exclusivement sur des tableaux de chiffres.

D'autres études menées par des publicitaires ont montré que dans le cas d'une exploration rapide d'un support et d'une information quelconque, la stratégie d'exploration de l'utilisateur fait un "z" : c'est à dire que l'oeil part du coin supérieur gauche, balaye une ligne de gauche à droite traverse l'image en diagonale du coin supérieur droit vers le coin inférieur gauche et balaye la dernière ligne de gauche à droite. Ce résultat n'a de sens que pour des utilisateurs qui ne connaissent pas la structure de l'information qu'ils vont

rencontrer sur l'écran et qui ont par contre des contraintes de temps.

Ces résultats ne sont pas parfaitement concordants puisqu'ils ne portent pas sur le même protocole d'expériences, mais ils ont en commun trois éléments :

- l'exploration commence par le coin supérieur gauche
- la zone centrale est parcourue systématiquement (en partant du coin supérieur gauche ou droit)
- l'extrémité gauche ou droite des images est mal vue.

Nous pouvons déduire de ces éléments que pour des utilisateurs novices ou occasionnels de l'informatique, les informations importantes (qui doivent être nécessairement lues) doivent commencer dans le coin supérieur gauche ou se situer dans la zone centrale.

Recherche sélective

Lorsque l'écran affiche une structure de l'information qui est connue de l'utilisateur et que celui-ci perçoit cette information dans un certain but, la recherche de l'information est sélective et non plus systématique; autrement dit, *la saisie de l'information est gouvernée par les processus cognitifs* . De manière plus précise, la saisie visuelle dépend moins des caractéristiques physiques du stimulus (intensité, taille, localisation, formes) que des stratégies opératoires de l'utilisateur, qui sont déterminées par les caractéristiques de la tâche et le degré d'expérience des utilisateurs.

Nous illustrons ces idées par une étude faite par Bouju et Sperandio (Bou,79) et portant sur les contrôleurs aériens.

L'objectif de cette étude était de voir si il y avait des différences de stratégie d'exploration visuelle selon le degré d'expérience des contrôleurs et selon l'augmentation du trafic aérien.

En règle générale, quand le trafic augmente on observe une prise d'information plus sélective, c'est-à-dire centrée sur les informations jugées pertinentes par les contrôleurs pour leur décision; les informations jugées pertinentes diffèrent selon le degré d'expérience des utilisateurs. Ce phénomène s'accompagne d'une augmentation de la durée moyenne de fixation sur les éléments pertinents.

Dans tous les cas, on n'observe pas une exploration systématique des informations, mais une fixation sélective sur les points où le contrôleur sait qu'il va trouver l'information pertinente.

Quand le volume du trafic augmente, le contrôleur expérimenté choisit un mode de régulation plus standardisé, c'est-à-dire plus économique pour lui.

Si on veut transposer ces résultats dans le cadre de l'informatique des organisations, on peut dire que l'utilisateur expérimenté ne va pas explorer l'ensemble de l'écran, mais uniquement les points qui lui semblent pertinents et qui pourraient ainsi être regroupés dans les zones observées le plus spontanément.

On remarque ainsi qu'en cas de surcharge de travail, l'utilisateur s'achemine vers une standardisation des procédures d'exploration. On facilitera le travail de l'utilisateur si, d'une application à l'autre, on lui fournit les informations de même nature dans les mêmes zones (standardisation des emplacements des zones de menus, messages d'erreurs, etc....).

Le concept d'homogénéité permettant d'acquérir des automatismes qui accélèrent la prise d'information, prime sur celui d'emplacement; peu importe que la barre de menu soit sur la ligne du haut ou du bas pourvu qu'elle soit toujours à la même place dans toutes les applications.

Les présentations sur écran

Nous présentons maintenant les différences de recommandations entre présentation des données et présentation des commandes.

La *présentation des données* peut se faire de deux manières différentes: par question/réponse ou par remplissage de formes.

Le mode question/réponse ne donne aucune latitude à l'utilisateur, il ne doit donc être utilisé que dans le cas où le guidage total par l'ordinateur s'impose.

Le remplissage de formes est plus souple car il peut permettre à l'utilisateur d'entrer les données dans un ordre différent de celui proposé (ce qui est très intéressant dans le cas d'un dialogue avec un client) mais aussi d'imposer un ordre dans le cas de saisie de formulaires.

Dans ce cas (saisie de formulaires), il est impératif que l'ordre de présentation des données soit le même que celui du formulaire.

La *présentation des commandes* peut ne pas exister dans le cas d'utilisation d'un langage de commande; cette absence de présentation ne peut convenir qu'à un public très spécialisé utilisant fréquemment ces logiciels.

Dans le cas usuel la présentation des commandes se fait sous forme d'un menu qui peut présenter les variabilités suivantes:

- visualisation uniquement à la demande (touche "aide")
- barre de menu affichée en permanence
- menu déroulant dont seul l'en-tête est visualisée, le reste apparaissant à la demande
- menu glissant qui permet de visualiser une liste de commandes plus grande que la place disponible
- menu dynamique qui signale à l'utilisateur les commandes disponibles à un moment donné de l'interaction (au moyen d'une surbrillance par exemple)

L'ensemble de ces présentations par menu demande peu de mémorisation de la part des utilisateurs et est donc particulièrement adapté à tous les utilisateurs novices et intermittents.

Selon les types de logiciel de gestion d'écran (voir chapitre 4), la présentation des données et des commandes peut se faire sous la forme de *monofenêtre* ou de *multifenêtre* ; la présentation monofenêtre est la plus répandue car la plus ancienne, mais la présentation multifenêtre est en plein développement.

L'avantage essentiel de cette dernière est de permettre la visualisation de plusieurs opérations simultanément, ce qui est d'une utilité certaine pour l'utilisateur dans le cas d'interruption et de transfert d'informations entre opérations. On peut même dire que ces deux types d'opérations ne sont de fait praticables qu'avec le multifenêtrage.

Le découpage de l'écran en fenêtres et le recouvrement partiel ou total peut être géré automatiquement par le logiciel ou être laissé à l'initiative de l'utilisateur; il n'y a pas de recommandations ergonomiques à ce sujet car cela dépend du type d'application et du type d'utilisateur.

Ici aussi, une expérimentation sur prototype peut s'avérer utile.

3.2.2.5 Le temps de réponse

La question qui se pose est la suivante : "Quel est le temps de réponse idéal?"

- immédiat, répondent Sperandio et Volleau (Spe,80) ;
- pas trop rapide, dit Wisner (Wis, 79), car cela crée une surcharge mentale par accélération des processus cognitifs.

Pour tenter de répondre plus précisément, nous pouvons nous référer à deux paramètres importants :

- dans une conversation entre deux personnes, le temps de réponse maximum est de l'ordre de deux secondes (étant bien entendu que la réponse peut être verbale ou représentée par un geste, un mouvement facial ou autre). D'après les observations menées, deux secondes semble bien correspondre à un seuil au-delà duquel l'individu a l'impression d'attendre ;

- par ailleurs, nous avons vu que les informations qui entrent ou qui sortent du cerveau humain transitent par une mémoire à court terme pendant 2 à 6 secondes au maximum.

En conséquence, si, en cours de travail, nous sommes interrompus par un événement extérieur, nous risquons d'effacer l'information de la mémoire à court terme.

En résumé, il est gênant d'être interrompu pendant la phase de mémorisation et il est pénible de devoir conserver trop longtemps l'information dans la mémoire à court terme, ce qui est le cas si le temps de réponse est supérieur à 4 secondes.

Par contre, un temps de réponse plus long pour une opération qui ne demande pas de mémorisation (l'impression, fin de travail), ne pose pas de problème à condition toutefois que l'utilisateur soit averti par un message affiché sur l'écran. Si le temps de réponse est vraiment trop long, il faut afficher ce temps pour que l'utilisateur puisse faire éventuellement autre chose en attendant. Sinon, cela peut provoquer de l'anxiété car il n'est pas en mesure de savoir si ce temps d'attente provient de contraintes techniques ou d'une erreur de sa part. Toujours pour la même raison d'anxiété, il faut éviter les temps de réponse trop variables.

Nous arrivons donc aux résultats suivants:

- moins de 2 secondes : temps de réponse idéal ;
- 2 à 4 secondes : impression d'attente, peu gênante pour la mémoire à court terme,
- plus de 4 secondes : trop long si le dialogue nécessite une mémorisation à court terme et s'il n'y a pas de messages affichés à l'écran.

3.2.2.6 Le traitement des erreurs

On distingue deux types d'erreur:

- les *erreurs d'exécution* qui proviennent du fait que l'on a tapé par inadvertance sur une autre touche que celle désirée; ce type d'erreur est souvent facilement détectable et rectifiable;

- les *erreurs d'intention* qui correspondent à une mauvaise interprétation du sens des commandes ou de la signification des procédures; ces erreurs peuvent n'être détectées que tardivement et leurs rectifications demandent un apprentissage de la part de l'utilisateur.

Pour le *signalement* des erreurs, il y a unanimité sur les éléments suivants:

- les erreurs doivent être signalées immédiatement ou le plus rapidement possible à cause de la "volatilité" de la mémoire à court terme,

- les messages d'erreur doivent être explicites et décrire, si possible, la cause de l'erreur,

- les messages d'erreur doivent être présentés toujours sous la même forme pour des raisons d'homogénéité et donc d'optimisation de la recherche sélective.

Par contre, il y a plusieurs possibilités pour l'emplacement proprement dit du message d'erreur :

- à côté de l'erreur, on est sûr que l'utilisateur verra le message mais il peut ne pas y avoir assez de place, sauf si l'on utilise une fenêtre en surimpression (logiciel de multifenêtrage)

- dans une zone spéciale on peut ne pas remarquer le message d'erreur, ce qui peut amener à rajouter des marques spéciales comme un clignotement du curseur ou de la zone, un signal sonore, une inversion vidéo...

Pour la *correction* des erreurs, l'utilisateur doit pouvoir revenir aisément à l'opération ou à la ligne où se situe l'erreur et il doit pouvoir annuler éventuellement en totalité ou en partie le travail qu'il a fait depuis lors.

Les possibilités de correction d'erreur posent des problèmes souvent complexes en informatique; ce ne sont pas des paramètres de surface du logiciel, mais des paramètres de la structure profonde que l'on définit lors de la Représentation Conceptuelle, en particulier quand on définit le séquençement des opérations.

3.2.2.7 Le guidage

Selon Senach (Sen,87), il y a essentiellement deux types de guidage:

- le guidage fonctionnel par description des commandes; ce guidage fournit pour chaque commande sa définition et ses effets, ses conditions d'exécution, ses effets de bord éventuels; ces descriptions peuvent être faites à trois niveaux de détails (résumé, détaillé, catalogue de problèmes)

- le guidage d'utilisation par reconnaissance des plans d'action de l'utilisateur; ce guidage suppose une métaconnaissance sur les différents buts possibles de l'utilisateur, une gestion du contexte de l'interaction (historique des transactions), une gestion du contexte de l'utilisateur (procédures adaptées à un utilisateur donné)

Le guidage fonctionnel est adapté à des utilisateurs expérimentés qui veulent se rappeler la syntaxe et la sémantique d'une commande.

Le guidage d'utilisation convient mieux à des utilisateurs novices ou intermittents car il peut les guider dans la réalisation de leur tâche en leur indiquant l'ensemble des opérations à parcourir en fonction du but qu'ils veulent atteindre.

3.3 La Représentation Externe

Nous avons vu que la Représentation Conceptuelle est constituée de deux ensembles de paramètres que nous avons appelés :

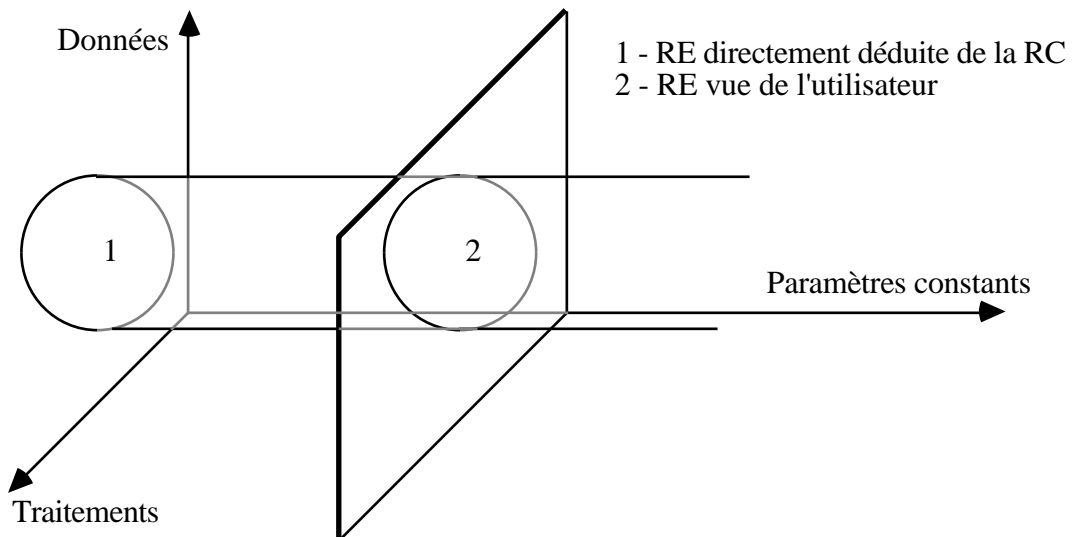
- les paramètres constants,
- les paramètres variables propres à une application particulière.

Ces deux ensembles ne sont pas du tout de même nature.

Les paramètres constants sont à mettre en oeuvre pour faciliter le travail de l'utilisateur quel que soit le type de tâches qu'il a à accomplir ce qui nécessite donc la conception d'outils généraux.

Les paramètres variables sont des éléments (traitement et données) issus de l'analyse du travail, spécifiques à chaque application.

A cause de cette différence, nous examinerons séparément la traduction de ces paramètres en R.E.



La définition des paramètres constants est totalement indépendante de celle des paramètres variables, mais l'interface résultante "vue" par l'utilisateur (R.E) sera la conjonction de ces deux ensembles d'éléments.

3.3.1 Les paramètres standard de la Représentation Externe

Dans ce paragraphe, nous décrivons tous les éléments qui ne sont pas liés à une application particulière; il s'agit, d'une part de la traduction des paramètres constants provenant de la Représentation Conceptuelle, d'autre part des options standard qui doivent être prises pour homogénéiser toutes les applications.

3.3.1.1 Traduction des paramètres constants

Nous avons défini trois types de paramètres constants qui sont ceux :

- d'aide au travail
- d'aide à l'apprentissage
- d'évolution.

Pour l'aide au travail, nous avons défini les éléments suivants :

- les possibilités d'interrompre à tout moment une tâche pour aller en exécuter une autre
- les possibilités de quitter le travail à n'importe quel moment
- les possibilités de différer le travail en cours
- la possibilité de transférer les données d'une tâche à l'autre.
- l'annulation pour la gestion des erreurs

- la mémorisation de l'activité de l'utilisateur (gestion du contexte) qui comprend quatre aspects (opérations interrompues, opérations différées, traces du travail, paramètres propres à l'utilisateur).

L'aide à l'apprentissage concerne le guidage fonctionnel et le guidage d'utilisation.

Les possibilités d'évolution concernent les éléments de la représentation conceptuelle que l'utilisateur peut modifier lui-même.

Ces paramètres sont qualifiés de constants en ce sens qu'ils sont utiles quelle que soit l'application traitée. Aussi, serait-il logique qu'ils soient intégrés dans un logiciel de base sur lequel on pourrait greffer les éléments propres à une application particulière.

C'est ce qui est fait, partiellement, avec les logiciels de multifenêtrage que nous décrivons au chapitre 4. Ils gèrent automatiquement l'interruption et le transfert de données entre opérations, les autres paramètres devant être programmés.

Nous pouvons constater que ces paramètres constants permettent de prendre en compte les recommandations ergonomiques concernant le séquençement des opérations, le guidage et partiellement la correction d'erreurs.

La question que nous devons résoudre ici est la suivante:

"Comment présenter ces possibilités à l'utilisateur au niveau de l'interface?"

La règle qui s'impose en priorité est celle de l'*homogénéité* ; ces paramètres étant communs à toutes les applications doivent être présentés exactement de la même façon quelle que soit l'application, c'est-à-dire qu'ils doivent:

- recevoir la même dénomination
- être présentés visuellement sur l'écran de la même manière
- être sélectionnés avec le même dispositif d'entrée.

Nous détaillons ci-dessous les critères que l'on peut mettre en oeuvre pour traduire les paramètres constants dans une Représentation Externe adaptée aux utilisateurs.

A titre d'illustration, nous présentons deux possibilités selon que l'on dispose d'un logiciel de gestion d'écran mono ou multifenêtré.

Subdivision des commandes

Dans le cas où nous mettons en oeuvre l'ensemble des possibilités énumérées précédemment ou, du moins, une partie importante

d'entre elles, il est nécessaire de procéder à un regroupement en sous-ensembles afin de ne pas surcharger l'écran.

Le critère de regroupement essentiel est la fréquence d'utilisation.

A titre d'exemple, l'ensemble des commandes standard est subdivisé en quatre sous-ensembles selon leur fréquence d'utilisation; cette subdivision est indépendante du type de logiciel de gestion d'écran:

- le premier sous-ensemble concerne l'aide à l'utilisation avec les commandes Interrompre Différer Quitter Annuler Copier Insérer
- le second sous-ensemble concerne la mémorisation avec la commande Mémoire à un premier niveau et les commandes Interruption Transaction Différé Utilisateur au second niveau
- le troisième sous-ensemble concerne l'aide à l'apprentissage avec la commande Guidage au premier niveau et les commandes Buts et Commandes au deuxième niveau
- le quatrième sous-ensemble concerne les possibilités d'évolution avec la commande Evolution au premier niveau et les commandes Procédure Opération Statut Déclenchement Précédence au deuxième niveau.

Dénomination des commandes

Le choix de la dénomination des commandes standard comporte un aspect arbitraire, car on ne peut se référer à aucun vocabulaire de spécialiste; les commandes standard correspondent à des opérations spécifiques de l'interaction homme-machine qui n'ont pas d'équivalent dans le travail manuel.

Mais, on peut s'inspirer des principes suivants:

- pour éliminer l'indétermination syntaxique, il y a lieu de nommer toutes les commandes sous la même forme syntaxique; on prendra soit la forme verbale soit la forme nominale. La pratique des informaticiens consiste plutôt à prendre la forme verbale, sans que cette pratique soit confirmée ou infirmée par des tests d'ergonomie;
- la dénomination de la commande doit refléter le point de vue des utilisateurs et non des informaticiens. Par exemple, la même commande s'appellera "consulter thesaurus" du point de vue de l'utilisateur et "afficher thesaurus" du point de vue de l'informaticien.

En cas d'ambiguïté, il est utile de procéder à des tests d'apprentissage et de mémorisation.

Disposition spatiale

La présentation des commandes standard à l'écran est guidée par deux critères:

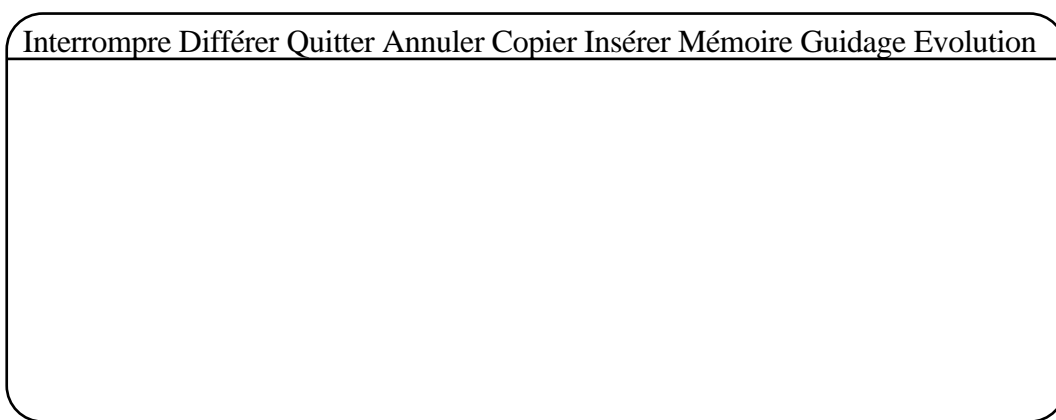
- ne pas surcharger l'écran de commandes inutiles ou utilisées peu fréquemment,
- rester au maximum homogène entre toutes les applications

Exemple avec logiciel monofenêtre:

Afin de ne pas surcharger l'écran, nous avons décidé que le menu standard n'utiliserait qu'une ligne ce qui, dans le cas usuel, correspond à 80 caractères. Ce choix nous oblige à créer deux niveaux de commandes où le premier niveau est affiché en permanence dans la ligne menu; le deuxième niveau n'apparaissant qu'à la demande.

Pour minimiser les manipulations, nous mettons dans le deuxième niveau les commandes utilisées plus rarement.

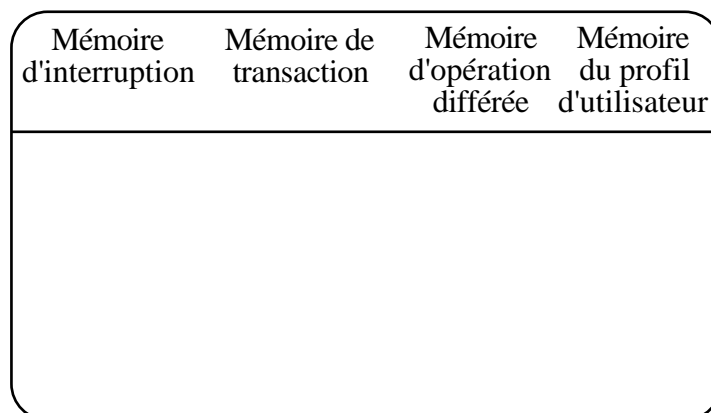
Nous aurons donc en permanence la barre de menu suivante:



Nous choisissons de disposer cette barre sur la première ligne de l'écran; nous aurions pu décider de la mettre sur la dernière ligne, l'essentiel étant de rester homogène sur l'ensemble des applications.

Si l'utilisateur sélectionne une des six premières commandes, il y aura exécution immédiate alors que les trois dernières commandes donneront lieu à l'affichage d'une autre barre de menu au même emplacement.

La commande Mémoire affichera la barre suivante:

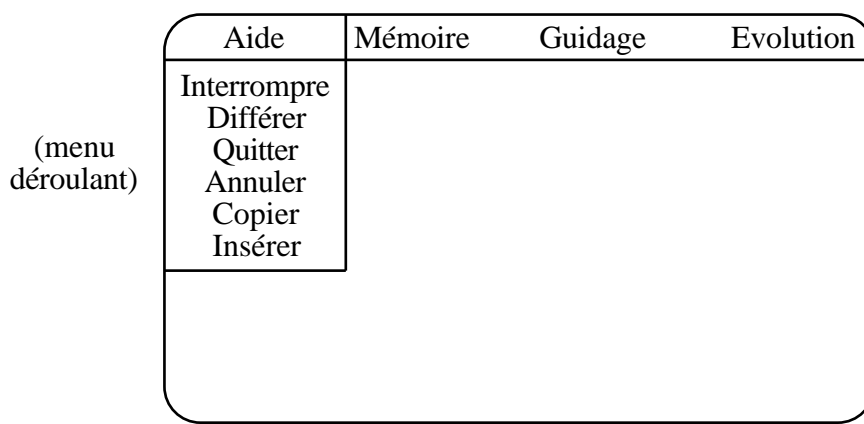


Exemple avec logiciel de multifenêtrage:

Le multifenêtrage permet d'utiliser un *menu déroulant* c'est à dire un menu qui apparaît dans une fenêtre en surimpression uniquement pendant le temps souhaité par l'utilisateur. L'avantage de ce type de menu réside dans le fait que l'on ne fait apparaître que ce qui concerne l'utilisateur et ceci sans surcharger l'écran.

Cela suppose une structuration des commandes en deux niveaux; les commandes du premier niveau étant affichées en permanence dans une barre de menu; les commandes du second niveau associées à chaque commande du premier niveau n'apparaissant qu'à la demande dans des menus déroulants.

Au premier niveau, nous indiquons les quatre sous-ensembles décrits dans le premier exemple sous la forme d'une barre de menu qui occupe la première ligne de l'écran, ce qui permet au menu déroulant d'apparaître au-dessous de chaque commande; si l'utilisateur sélectionne la commande Aide, il obtiendra:



Et il obtiendrait de la même manière les commandes de deuxième niveau associées à Mémoire Guidage et Evolution.

Sélection des commandes

Pour sélectionner les commandes, on peut utiliser diverses possibilités:

- positionner le curseur sur la commande au moyen des touches-fonctions de déplacement (ou d'une souris,...) puis valider par une touche-fonction. Cette possibilité est intéressante pour les utilisateurs novices ou intermittents car elle demande peu d'effort de mémorisation;
- taper la (ou les) première lettre de la commande en association avec une autre touche indiquant que l'on est en mode saisie de commande. Ce mode de désignation est plus complexe à mémoriser que le précédent; mais il peut être utilisé par des non-spécialistes, si les commandes et leurs abréviations sont affichées en entier et en permanence sur l'écran. Dans le cas contraire, il faut le réserver à des spécialistes assidus;

- utiliser des touches-fonctions associées à chaque commande. Cette possibilité est très efficace pour des utilisateurs expérimentés.

Exemple:

Nous choisissons d'implémenter deux possibilités ; la première citée pour les utilisateurs novices et la dernière pour les utilisateurs expérimentés.

Pour des utilisateurs novices ou intermittents, la sélection la plus simple d'une commande se fait au moyen de la souris (ou de toute autre désignation directe); quand l'utilisateur positionne avec la souris le curseur sur la commande Aide le menu ci-dessus apparaît tant que l'utilisateur appuie sur le bouton de la souris; il ne lui reste plus qu'à faire glisser le curseur jusqu'à la commande désirée et à relâcher le bouton de la souris sur cette position pour que cette commande soit sélectionnée. Cette manoeuvre est en fait plus difficile à décrire par écrit qu'à exécuter physiquement!

Pour des utilisateurs plus expérimentés qui n'ont pas besoin de visualiser le menu déroulant, il est plus rapide de taper sur une touche-fonction .

Traitement des erreurs

On peut détecter deux types d'erreur, soit l'utilisateur sélectionne une commande qui n'existe pas, soit il sélectionne une commande qui existe mais qui ne peut être activée à ce moment là (c'est le cas de la commande Copier qui ne peut être sélectionnée deux fois de suite sans risque d'ambiguïté).

Dans le premier cas (commande inexistante), on fait apparaître un message d'erreur dans une zone de l'écran spécialisée à cet effet selon les options définies dans l'écran standard (voir paragraphe suivant).

Avec le multifenêtrage, la visualisation des messages d'erreur peut se faire dans une fenêtre en surimpression ce qui a l'avantage d'être très bien perçu par l'utilisateur.

Dans le second cas, on peut procéder de la même manière mais il est plus efficace d'indiquer également sur les commandes elle-mêmes celles qui sont activables et celles qui ne le sont pas afin de minimiser ce type d'erreur; ceci correspond à la notion de menu dynamique, c'est-à-dire au signalement a priori des commandes possibles à un moment donné.

Pour présenter ce menu dynamique, on fait ressortir visuellement (surbrillance, inversion vidéo,...) les commandes activables à un moment donné par rapport aux commandes non activables; on peut également faire disparaître de l'écran les commandes non activables, mais cette façon de procéder est moins judicieuse car elle ne préserve pas l'homogénéité de la présentation.

3.3.1.2 Les options standard

Disposition spatiale

Avant de décrire une opération et une procédure particulière, il est impératif de déterminer *une disposition standard de l'écran* qui servira de cadre pour l'ensemble des applications.

Les critères de disposition doivent s'inspirer des résultats d'ergonomie sur la recherche systématique et sélective; nous avons vu qu'un novice va balayer l'écran en "z" , alors qu'un expérimenté consultera directement la zone qui l'intéresse (si l'écran est standard, cette stratégie devient très efficace).

Aussi, on peut en déduire que, si on veut que les zones de menu (dans la mesure où elles existent) soient vues par des utilisateurs novices, il faut les mettre sur la première ou la dernière ligne de l'écran; cette option ne constitue pas une gêne pour les utilisateurs expérimentés dans la mesure où ils ne consulteront cette zone que s'ils en ont besoin.

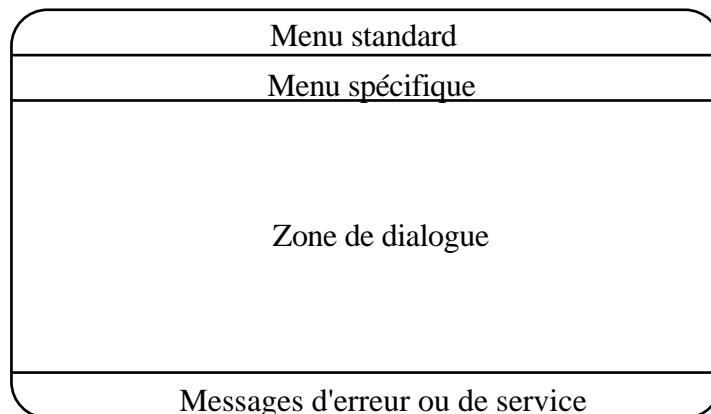
Un autre critère à utiliser est la non surcharge de l'écran; ce critère peut être en contradiction avec les autres car il peut amener à ne pas spécialiser des zones d'écran, de manière à libérer tout l'écran pour la zone de dialogue dans le cas d'opérations à très forte densité d'informations.

Les possibilités de disposition sont bien sûr différentes selon que nous disposons ou non d'un logiciel de multifenêtrage; nous examinons à titre d'exemple les deux possibilités.

Exemple de disposition standard monofenêtre :

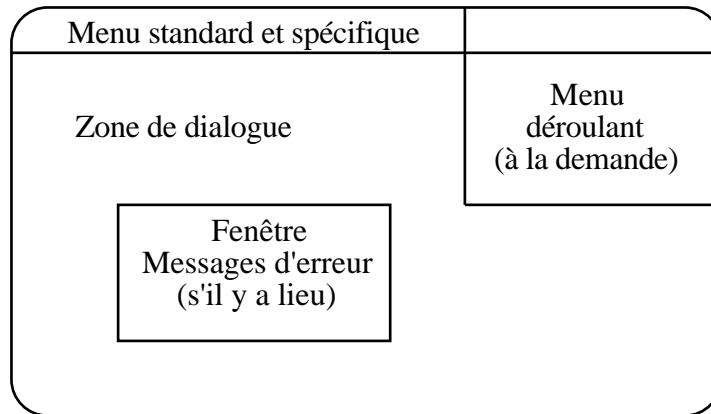
Nous spécialisons quatre zones:

- les zones de menu spécifique et standard occupent les deux premières lignes de l'écran
- la zone de message d'erreur et de service occupe la dernière ligne
- la zone de dialogue utilisée dans chaque opération occupe tout le reste de l'écran.



Exemple de disposition standard multifenêtre :

- les zones de menu spécifique et standard n'occupent que la première ligne de l'écran car l'utilisation de menu déroulant permet de regrouper plus fortement les commandes sans gêne pour l'utilisateur
- la zone de message d'erreur et de service n'existe pas en permanence; elle sera éditée dans une fenêtre en surimpression qui ne sera affichée qu'en cas de message
- la zone de dialogue occupe le reste de l'écran.



Syntaxe

Une partie de la syntaxe du langage d'interrogation a un caractère général et doit donc être déterminé à ce niveau de manière à être homogène sur l'ensemble des applications. Il s'agit:

- du mode de sélection des commandes (standard ou spécifiques)
- de la validation des données ou des écrans.

Messages de service

Le contenu des messages de service est dépendant du contenu d'une application particulière mais il existe un message de service à caractère général, c'est celui concernant le temps de réponse.

En effet, le temps de réponse n'est pas connu à ce stade de la conception puisqu'il dépend de la Représentation Interne de l'application, mais nous avons vu qu'au-delà de 2 secondes l'utilisateur a l'impression d'attendre et qu'au delà de 4 secondes, il est indispensable de lui signaler par un message que la procédure se déroule normalement.

Le contenu et la forme du ou des messages concernant le temps de réponse doivent être déterminés à ce stade car ils sont indépendants du contenu de l'application.

3.3.2 Les paramètres de la Représentation Externe spécifiques à l'application

Quand l'étape de la Représentation Conceptuelle est finie, nous disposons, pour chaque but d'un poste de travail, de spécifications complètes concernant la structure de données et la structure de traitement des opérations de la procédure.

Pour chaque opération interactive, nous connaissons:

- la liste de ses actions
- la liste de ses entrées
- la liste de ses sorties
- ses pré-requis
- son déclenchement

Pour chaque procédure, nous avons:

- la liste des opérations activables
- les précédences entre ces opérations.

Au niveau de la Représentation Externe, il nous reste à définir pour chaque opération interactive:

- la disposition du ou des écrans
- le vocabulaire des données
- le contenu des messages d'erreur, de service ou de guidage
- la syntaxe
- les dispositifs d'entrée.

Pour l'ensemble de la procédure, nous devons définir:

- les noms des commandes activant les opérations
- la hiérarchie du menu
- la présentation du menu
- la syntaxe et les dispositifs d'entrée
- les messages d'erreur et de guidage.

Nous illustrons cette démarche sur un exemple qui reprend celui de la documentation développé dans le chapitre deux. Nous décrivons l'opération "enregistrement des bases du document" et la procédure minimale d'un poste de documentation.

3.3.2.1 Représentation Externe des opérations

Vocabulaire

Pour la détermination du vocabulaire des données, il n'existe qu'un seul critère qui est celui d'utiliser le vocabulaire des spécialistes de la tâche (ou le vocabulaire courant pour les applications "grand public").

Dans les cas (rares) où il y a création de données qui n'existent pas dans les tâches actuelles, il est nécessaire de procéder à des tests d'apprentissage et de mémorisation pour ces nouveaux noms.

Exemple:

Pour décrire les données qui permettent au documentaliste d'enregistrer un document nous choisissons le vocabulaire déjà utilisé par le documentaliste qui est: COTE, AUTEUR, TITRE, EDITEUR, ANNEE, NOMBRE DE PAGES, LANGUE.

Syntaxe

L'indication de fin de saisie de données doit être faite par le même moyen sur l'ensemble de l'application.

Le même principe doit être appliqué pour la validation de l'ensemble de l'écran.

Disposition spatiale

Si l'opération correspond à un document sur papier ou à un écran existant déjà, il faut, dans la mesure où ces documents ou ces écrans sont adaptés aux nécessités de la tâche, s'en inspirer fortement compte-tenu des nouvelles contraintes techniques.

Dans les autres cas, il faut rechercher la ou les logiques d'utilisation de cette opération de manière à présenter les données dans l'ordre de la logique d'utilisation la plus courante en prévoyant éventuellement des possibilités de variantes dans l'ordre de remplissage, par exemple, s'il y a lieu.

Exemple:

Nous reproduisons sur l'écran l'ordre et la disposition spatiale contenu dans une fiche documentation manuelle car les documentalistes y sont habitués et surtout parce que les systèmes manuels et automatiques vont continuer à coexister, ce qui donne:

Zone de dialogue

The diagram shows a rounded rectangular box containing a form with the following fields and labels:

- Cote
- Année
- Titre
- Auteur(s)
- Editeur
- Nombre de pages Langue

Dispositif d'entrée

Pour une opération, le dispositif d'entrée concerne la saisie des données et l'enchaînement des différentes zones de saisie.

Les données peuvent être saisies au clavier, ou sélectionnés sur une liste affichée par le logiciel, ou en provenance d'une autre opération.

Le premier cas ne pose pas de problème; dans les deux autres cas, il faut veiller au fait que les modes de désignation soient homogènes sur l'ensemble de l'application.

Le mode d'enchaînement des zones de saisie est déterminé dans la Représentation Conceptuelle; s'il est à l'initiative de l'ordinateur, cela se traduit par un positionnement automatique du curseur; s'il est à l'initiative de l'utilisateur, il faut prévoir un mode de désignation d'une zone par l'utilisateur. Le plus simple et le plus efficace est la désignation directe (souris, light pen,...) ou à la rigueur les touches-fonction de déplacement de curseur.

Exemple:

Les zones sont enchaînées automatiquement car cet ordre correspond aux habitudes des documentalistes et tous les renseignements sont groupés sur un même document. Le choix des touches-fonctions pour la syntaxe est motivé par le fait qu'il y a beaucoup de saisie au clavier et que les documentalistes sont très habitués à utiliser un clavier.

Messages d'erreur

Nous devons déterminer l'ensemble des erreurs que l'utilisateur peut commettre et surtout que le logiciel peut détecter afin de prévoir les messages d'erreurs correspondants. Chaque message d'erreur doit être le plus explicite possible et indiquer le type de contrôle qui est effectué.

Exemple:

Sur la "Cote" il peut y avoir deux types d'erreur:

- une erreur de vraisemblance si l'utilisateur tape une cote incompatible avec la composition prévue (deux lettres suivies au maximum de six chiffres) auquel cas le message sera "mauvaise composition de la cote";
- une erreur de concordance si la cote correspond à une cote existant déjà ce qui est impossible puisque cette opération a pour but de permettre l'enregistrement de nouveaux livres; le message d'erreur étant "cote déjà attribuée".

Sur l' "Année" il peut y avoir un contrôle de vraisemblance sur une fourchette minimum/maximum avec le message "erreur sur la date".

Sur l' "Auteur" il ne peut pas y avoir de vérifications.

Sur l' "Editeur", il y a un contrôle de concordance avec une liste d'éditeur et le message est donc "éditeur non enregistré".

Sur le "Nombre de pages", il y a un contrôle de vraisemblance sur les chiffres avec en réponse le message "le nombre de pages doit être numérique".

Sur la "Langue", il y a un contrôle de concordance par rapport à une liste de langues avec le message "langue non enregistrée".

3.3.2.2 Représentation Externe des procédures

Vocabulaire

Nous devons déterminer l'ensemble des noms de commande correspondant à chaque opération. Là aussi, il faut au maximum adopter le vocabulaire employé par les spécialistes de la tâche, mais ce n'est pas toujours possible quand des opérations automatisées ne correspondent pas à des opérations manuelles.

Dans ce cas, on s'inspirera des deux principes développés pour la dénominations des commandes standard (forme verbale ou nominale, dénomination du point de vue de l'utilisateur); en cas d'ambiguïté on procédera à une expérimentation.

Exemple:

Nous donnons ci-dessous la liste des différentes opérations de la procédure minimale d'un poste de documentation et la liste correspondante des noms de commande.

<i>Liste des opérations de la procédure</i>	<i>Liste correspondante des commandes</i>
Enregistrer les bases du document	Indexer (vocabulaire du spécialiste)
Créer termes du thesaurus	Créer mots-clés
Créer liens entre termes	Lier mots-clés
Consulter thesaurus	Consulter thesaurus
Rechercher documents	Chercher document
Imprimer thesaurus	Imprimer
Enregistrer mots-clés	-

L'opération "Enregistrement mots-clés du document" n'a pas de nom de commande correspondant car elle correspond à un déclenchement automatique dans la procédure minimale et ne peut donc être déclenchée par l'utilisateur.

Hiérarchie du menu

Dans le cas général, l'ensemble des commandes d'une application interactive est trop important pour pouvoir être présenté à l'utilisateur sous la forme d'une liste unique. Aussi, on subdivise l'ensemble des commandes de manière à obtenir une structure arborescente ou hiérarchisée; l'utilisateur doit alors parcourir cette arborescence pour activer les commandes utiles à son travail.

Cette structure hiérarchique du menu ne doit en aucun cas être faite par rapport à la logique des traitements informatiques mais par rapport à la logique d'utilisation des utilisateurs. Cette logique d'utilisation a été déterminé au niveau de la Représentation Conceptuelle quand on décompose le travail à effectuer en buts, sous-but, ..., opérations.

C'est cette structuration qui doit être reprise ici; les buts correspondent au niveau un de la structure arborescente, les sous-buts au niveau deux et les opérations au dernier niveau.

Exemple:

Comme nous l'avons vu au chapitre 2, à la suite d'une étude sur la logique d'utilisation des documentalistes, on a constaté que le but de la procédure qui est de gérer les documents se subdivise en deux sous-buts: "enregistrer document" et "consulter document".

Les commandes ci-dessus sont donc regroupées en deux sous-ensembles:

- le premier sous-ensemble nommé "Enregistrer" regroupe toutes les commandes permettant d'atteindre ce sous-but, soit Indexer, Créer mots clés, Lier mots clés, Consulter thesaurus.

- le second sous-ensemble nommé "Consulter" regroupe Consulter thesaurus, Chercher document, Imprimer.

Comme nous l'avons déjà remarqué, dans ce découpage en logique d'utilisation, certaines commandes (ici, consulter thesaurus) peuvent apparaître dans plusieurs sous-menus puisque l'objectif est de minimiser les manipulations de l'utilisateur.

Présentation du menu

Le menu peut être affiché en permanence ou seulement à la demande de l'utilisateur; l'affichage permanent est plus favorable aux utilisateurs novices ou intermittents alors que l'affichage à la demande, s'il est utilisé, doit être réservé aux utilisateurs expérimentés.

Dans le cas d'affichage permanent, la ligne où est affichée le menu a un contenu évolutif selon la demande de l'utilisateur; au premier niveau, elle contient la liste des buts et au dernier niveau, la liste des opérations.

Avec un logiciel de multifenêtrage, on peut avoir une variante qui consiste à mettre la liste des opérations dans un menu déroulant.

Exemple avec logiciel monofenêtre:

Dans la barre de menu initiale (au premier niveau), on affiche les sous-buts: Enregistrer et Consulter.

Quand l'utilisateur sélectionne un but, la barre de menu initiale est remplacée par l'ensemble des commandes dépendantes de ce but; par exemple, si l'utilisateur sélectionne "Enregistrer", on affiche:

Indexer, Créer mots-clés, Lier mots-clés, Consulter thesaurus.

L'utilisateur dispose ainsi de l'ensemble des commandes lui permettant de réaliser un enregistrement sans avoir à revenir au premier niveau .

Exemple avec logiciel de multifenêtrage:

La barre de menu initiale (contenant les buts) reste affichée en permanence et la liste des opérations dépendante d'un but apparaît à la demande de l'utilisateur dans un menu déroulant comme nous l'indiquons dans le schéma suivant:

Enregistrer	Consulter
Indexer Créer mots-clés Lier mots-clés Consulter thesaurus	

Dispositif d'entrée

Comme pour la sélection des commandes standard, on peut utiliser le clavier avec la touche "mode commande" et la frappe d'une abréviation de la commande désirée ou la souris en positionnant le curseur sur la commande et en cliquant ou une touche-fonction.

Exemple:

On réserve les touches fonctions aux commandes standard.

Pour les commandes spécifiques à l'application, on offre deux types de mode de sélection laissé au choix de l'utilisateur; la désignation directe par la souris et la combinaison d'une touche commande et d'une abréviation de la commande.

Messages d'erreur

Comme pour le menu standard, nous avons deux types d'erreur:

- l'erreur de concordance, si la lettre tapée ne correspond pas à une commande; il est important de remarquer qu'il ne peut pas y avoir d'erreur si la sélection se fait par cliquage de la souris.
- l'erreur d'utilisation si l'utilisateur veut activer une commande existante mais non activable à ce stade de la transaction.

Exemple:

Pour l'erreur de concordance, le message sera "commande inconnu, veuillez la retaper".

Le problème de l'erreur d'utilisation ne se pose pas dans cet exemple car, dans la procédure minimale du poste de documentation, il n'y a qu'une précedence qui est à déclenchement automatique. Par contre le cas pourrait se poser dans une autre procédure effective.

Guidage

Il est important ici de prévoir le contenu des messages qui serviront de guide à l'utilisateur.

Pour le guidage fonctionnel, on fournira pour chaque commande, un résumé décrivant le rôle et les effets de cette commande, éventuellement illustré par un exemple; on peut ensuite rajouter un

niveau d'explication plus détaillé avec une liste exhaustive de tous les problèmes que l'on peut rencontrer pour exécuter cette commande.

Pour le guidage d'utilisation, il faut exploiter la structuration en buts et sous-butts de la Représentation Conceptuelle puisqu'elle correspond à une logique d'utilisation selon les buts de l'utilisateur et qu'elle permet donc de répondre à des questions du type "comment faire pour...?".

De plus, ce guidage d'utilisation peut être personnalisé en utilisant une procédure effective propre à un utilisateur ainsi que la mémorisation des transactions déjà effectuées.

Synthèse de la Représentation Externe

La Représentation Externe telle qu'elle sera vue par l'utilisateur est une composition des paramètres standard et des paramètres variables de l'application puisqu'à chaque moment d'une transaction, plusieurs options sont offertes à l'utilisateur; ces options provenant à la fois des commandes standard et des commandes spécifiques à l'application.

Pour illustrer ce fait, nous présentons ci-dessous un exemple d'écran utilisant un logiciel de multifenêtrage avec interruption d'opération. L'opération "Indexer" a été interrompue (paramètre standard) de manière implicite par l'activation de l'opération "Consulter thesaurus" car le documentaliste voulait connaître les mots-clés reliés par une relation de hiérarchie au mot-clé "informatique" afin de mener à bien son opération d'indexation de document.

DOCUMENTATION		
Options standard	Enregistrer	Consulter
Indexer		
<i>Cote</i>	<input type="text" value="X235L"/>	<i>Année</i> <input type="text" value="1988"/>
<i>Titre</i>	<input type="text" value="Histoire de l'informatique"/>	
<i>Mots-clés</i>	<input type="text" value="Epistémologie"/>	<input type="text" value="Langages"/> <input type="text" value="Cybernétique"/>
<input type="text" value="Informatique"/>	Consulter thesaurus	
<input type="text" value="Réseaux"/>	<i>Mot-clé</i> <input type="text" value="Informatique"/>	<i>Type de lien</i>
	<i>Mots liés</i>	<input type="radio"/> Hiérarchie
	Informatique de gestion ▲	<input type="radio"/> Synonymie
	Langages	<input type="radio"/> Homonymie
	Analyse	<input type="radio"/> Voisinage
	Programmation	
	Réseaux de Petri ▼	
		<input type="button" value="Quitter"/>

Chapitre 4 :

STRUCTURE INTERNE DES LOGICIELS INTERACTIFS

Nous présentons dans ce chapitre la conception et la mise en oeuvre de la Représentation Interne de l'application interactive qui d'une part représente le point de vue du programmeur sur cette application, d'autre part est la traduction de la Représentation Conceptuelle et de la Représentation Externe.

Les possibilités de mise en oeuvre sont très différentes selon le type de logiciel de gestion d'écran dont on dispose.

On distingue les logiciels de gestion d'écran classiques gérant une seule fenêtre des logiciels plus élaborés permettant entre autre de gérer plusieurs fenêtres et que nous appellerons dans la suite de ce chapitre des systèmes de Gestion Multifenêtre ou SGM.

Nous ne détaillons pas ici les caractéristiques de logiciels de gestion d'écran classiques car ils sont bien connus des informaticiens. Nous dirons simplement qu'il en existe deux types; les plus élémentaires permettent uniquement de définir les sorties sur écran ligne après ligne alors que les plus évolués permettent de définir globalement un écran de manière interactive.

Par contre dans un premier paragraphe, nous présentons de manière détaillée les SGM (Weg,83) car ils sont encore peu utilisés alors qu'ils présentent des fonctionnalités très intéressantes pour la mise en oeuvre d'une Représentation Externe adaptée à l'utilisateur.

Dans le deuxième paragraphe, nous montrons comment mettre en oeuvre les Représentations Conceptuelle et Externe au niveau de la programmation.

4.1 Présentation des systèmes de gestion d'écran multifenêtre

La fonction principale d'un SGM est de permettre l'accès en parallèle à différentes applications et de leur donner la possibilité d'échanger des informations. Pour permettre ces accès parallèles,

l'écran est divisé en fenêtres et à chaque fenêtre est associée une application.

Le SGM. est responsable de la gestion de l'écran et des différents périphériques d'entrée du système (clavier, souris, ...).

L'intérêt d'un tel SGM. est de fournir au programmeur d'application un ensemble de primitives qui lui permettront d'intégrer dans les applications une gestion très évoluée des périphériques d'entrée/sortie sans qu'il ait besoin de programmer ses fonctionnalités à chaque fois.

Le gain pour le programmeur peut être comparé au passage des systèmes de gestion de fichiers aux Systèmes de Gestion de Bases de Données. Dans le cadre d'une application bâtie sur un système de gestion de fichiers le programmeur intègre la gestion des liens entre fichiers pour chaque application particulière alors qu'il en est déchargé avec un Système de gestion de Base de Données. De la même manière, un SGM.lui permet de se décharger des problèmes d'interruption et transfert de données entre applications.

Un SGM présente également un intérêt pour l'utilisateur car il lui apporte une grande souplesse dans la gestion du dialogue.

Nous présentons tout d'abord le SGM du point de vue de l'utilisateur, puis du point de vue du programmeur et, en dernier lieu, l'exemple d'un SGM particulier.

4.1.1 Les fonctionnalités d'un SGM vues par l'utilisateur

4.1.1.1 Liste des fonctions

Les fonctions principales d'un SGM. sont de permettre l'accès en parallèle à différentes applications et la communication entre applications.

La *communication entre applications* présente deux aspects :

- démarrage et arrêt des applications; il doit être possible d'activer de nouvelles applications à partir de celles déjà actives, sans qu'il soit nécessaire de désactiver les premières (interruption et reprise). Par exemple, l'utilisateur doit pouvoir visualiser les informations du fichier Clients alors qu'il fait une création.
- échange d'informations entre applications actives. Par exemple, l'utilisateur peut échanger une information du fichier Clients au document Facture.

En s'appuyant sur ces deux fonctionnalités (accès parallèle, communication), le SGM permet également d'effectuer la gestion des menus, des désignations, de l'écho, des erreurs, du signal et de l'aide à l'utilisation.

La gestion des menus

le menu (ensemble de commandes) autorisé à un moment donné de l'activité peut apparaître dans une fenêtre spéciale.

Il peut soit être affiché constamment à l'écran, soit apparaître sur l'écran à la demande de l'utilisateur et disparaître dès qu'une commande est choisie.

Le menu peut être modifié interactivement par les activités afin de ne donner que les commandes permises à l'utilisateur à un moment donné (notion de menu dynamique).

La gestion des désignations

La désignation permet à l'utilisateur d'identifier un point sur l'écran ; la sélection permet d'identifier une entité logique. Une sélection peut être définie par une ou plusieurs désignations; les sélections peuvent se faire de deux façons :

- soit en désignant l'information de façon spatiale, par exemple pour supprimer une phrase, on désigne le début et la fin de cette phrase;
- soit en utilisant la structure hiérarchique de cette information. Le parcours de cette hiérarchie peut se faire de façon ascendante (du mot au document), descendante (du document au mot) ou horizontale (information au même niveau dans différents documents).

La gestion de l'écho

L'idée est de renvoyer un écho (ou rétroaction) à l'utilisateur lui permettant de s'assurer que son action immédiate a bien été prise en compte par l'ordinateur.

S'il vient d'activer une commande, l'écho peut être tout simplement le résultat de l'opération (si celle-ci peut être faite immédiatement). On peut également signaler cette commande par un moyen visuel quelconque (surbrillance, inversion vidéo...). Il peut être également intéressant de signaler que l'ordinateur est en train de travailler (en cas de temps de réponse supérieur à 2 secondes) ou qu'il est en attente de paramètres de l'utilisateur.

Si l'utilisateur vient de sélectionner un objet, celui-ci peut être mis en évidence comme pour les commandes (clignotements, inversion vidéo...).

La gestion des erreurs

Le système gère deux fonctions :

- le signalement des erreurs soit dans une fenêtre dédiée soit près d'une zone erronée.

- les commandes de rectification d'erreurs. On peut trouver des commandes comme :
 - "annuler" pour une exécution en cours
 - "arrêter" pour arrêt temporaire d'exécution
 - "reprendre" après un arrêt temporaire
 - "défaire" pour se retrouver dans l'état précédent l'activation d'une commande.

La gestion du signal

Elle permet d'indiquer à l'utilisateur qu'un événement vient de parvenir à l'ordinateur. Par exemple, dans le cas de messages électroniques, on signalera par une icône à l'utilisateur l'arrivée d'un document dans une boîte aux lettres.

L'aide à l'utilisateur

A tout moment sur demande de l'utilisateur, le système peut expliquer une commande particulière:

- l'ensemble des fonctions accessibles à un moment donné
- la documentation complète du système
- les causes possibles des erreurs.

4.1.1.2 La mise en oeuvre de ces fonctions

Nous entendons par mise en oeuvre les différentes manières dont dispose l'utilisateur pour mettre en oeuvre le démarrage et l'arrêt du système et des activités, les fenêtres et les commandes

Démarrage et arrêt du système

Le système dispose d'un certain nombre de valeurs par défaut.

L'identifiant au démarrage de l'utilisateur peut permettre de personnaliser le système en ce sens qu'il activera les dernières valeurs par défaut fournies par cet utilisateur.

Un autre aspect de la personnalisation consiste à conserver le contexte des activités entre deux sessions de travail de manière que l'utilisateur puisse redémarrer dans l'état où il avait arrêté son travail.

Gestion des fenêtres

Les fenêtres sont utilisées par les différentes activités pour contenir des informations ou des menus. La fenêtre et l'application étant fortement liées, la mise en oeuvre la plus simple pour l'utilisateur consiste à créer automatiquement une fenêtre dès qu'il active une nouvelle activité et inversement à détruire automatiquement cette fenêtre lorsque cette activité est considérée comme terminée (fin de saisie de paramètres ou commande explicite de l'utilisateur).

En général, cette fenêtre est constituée de deux parties appelées clôture et cadre:

- *la clôture* est la partie de la fenêtre délimitant l'espace où peut être affecté l'information de l'activité. Si cette information est beaucoup plus grande que l'espace délimité par la clôture, on peut avoir une zone de défilement (ou un "page à page") commandée par l'utilisateur.

- *le cadre* entoure la clôture et permet de distinguer les fenêtres les unes des autres car il contient des informations permettant d'identifier la fenêtre et éventuellement des zones pour les commandes du menu.

La gestion des fenêtres implique un *partage de l'écran* qui peut être géré de deux manières :

- l'utilisateur choisit la taille et la position de ses fenêtres. Ces différentes fenêtres peuvent se recouvrir partiellement ou totalement. L'inconvénient est que l'utilisateur assure seul la gestion de l'écran;

- le système peut gérer complètement le partage de l'écran en optimisant l'occupation de l'écran et en interdisant le recouvrement de fenêtres. Plus on crée de fenêtres, plus la taille de chaque fenêtre diminue (on est de fait limité à 4 ou 6 fenêtres pour des raisons de lisibilité). L'utilisateur peut intervenir sur la grandeur respective des différentes fenêtres.

On peut bien sûr imaginer des variantes entre ces deux manières extrêmes. Par exemple, toute nouvelle fenêtre est créée automatiquement dans un espace libre (au besoin en diminuant la taille des autres fenêtres) et l'utilisateur peut ensuite demander un recouvrement partiel ou total, ce qui peut être utile dans le cas où un travail de fond est exécuté en parallèle avec d'autres activités.

Il peut exister des *liens hiérarchiques* entre fenêtres :

- lien mère-fille : une opération (fermeture, ouverture, déplacement) sur la fenêtre mère entraîne automatiquement la même opération sur la fenêtre fille;

- lien frère-soeur : deux fenêtres sont soeurs si elles sont filles d'une même fenêtre. L'utilisateur ne peut travailler que sur une des soeurs à la fois (si deux fenêtres soeurs occupent une même position dans l'écran, une des deux soeurs recouvre l'autre).

Le jeu minimum de commandes de manipulation de fenêtres est le suivant :

- création de fenêtre ; cette commande peut être automatique si elle est activée par la création d'une activité. La taille et la position peuvent être choisies automatiquement ou indiquées par l'utilisateur;
- changement d'activité courante (l'activité courante est l'activité vers laquelle les entrées du clavier sont dirigées);
- changement de position et d'ordre de superposition;
- changement de taille de la fenêtre; cette modification peut soit avoir un effet de "loupe" sur l'information visualisée, soit modifier la quantité d'informations affichées. Il peut être intéressant de pouvoir résumer une fenêtre à son titre.

La plupart des commandes sont offertes explicitement à l'utilisateur soit dans le cadre des fenêtres, soit dans une fenêtre réservée au gestionnaire de fenêtres.

Si le système a des commandes génériques (valables pour toutes les activités), ces commandes peuvent être utilisées pour manipuler les fenêtres. Par exemple, la même commande peut détruire un paragraphe d'un document et d'une fenêtre.

Certaines commandes peuvent être implicites. Nous avons déjà vu que la création d'une activité peut entraîner la création d'une fenêtre. De même, le changement d'activité courante peut se faire automatiquement quand l'utilisateur désigne une fenêtre.

Syntaxe des commandes

Nous pouvons avoir une syntaxe préfixée, postfixée ou infixée.

- Dans une commande préfixée, l'utilisateur indique le nom de la commande plus les paramètres de cette commande. Cette syntaxe est souvent considérée comme plus "naturelle" car plus proche de la manière de penser de l'utilisateur. Cette syntaxe place l'utilisateur dans un mode temporaire. En effet, après chaque commande il peut spécifier seulement les paramètres ayant trait à cette commande. Ceci est un inconvénient si l'utilisateur veut utiliser plusieurs commandes et si on veut conserver la simplicité de la syntaxe.

- Dans une commande postfixée, l'objet à traiter est spécifié en premier et il est suivi par l'opération à accomplir. Cette syntaxe, plus simple que la précédente pour les commandes à un paramètre, s'adapte mal aux commandes à plusieurs paramètres, si on veut conserver la simplicité de syntaxe.

- La syntaxe infixée est intermédiaire entre la postfixée et la préfixée. Les paramètres entourent la commande, ce qui rend simple une commande à deux paramètres, sinon on cumule les inconvénients des deux autres syntaxes.

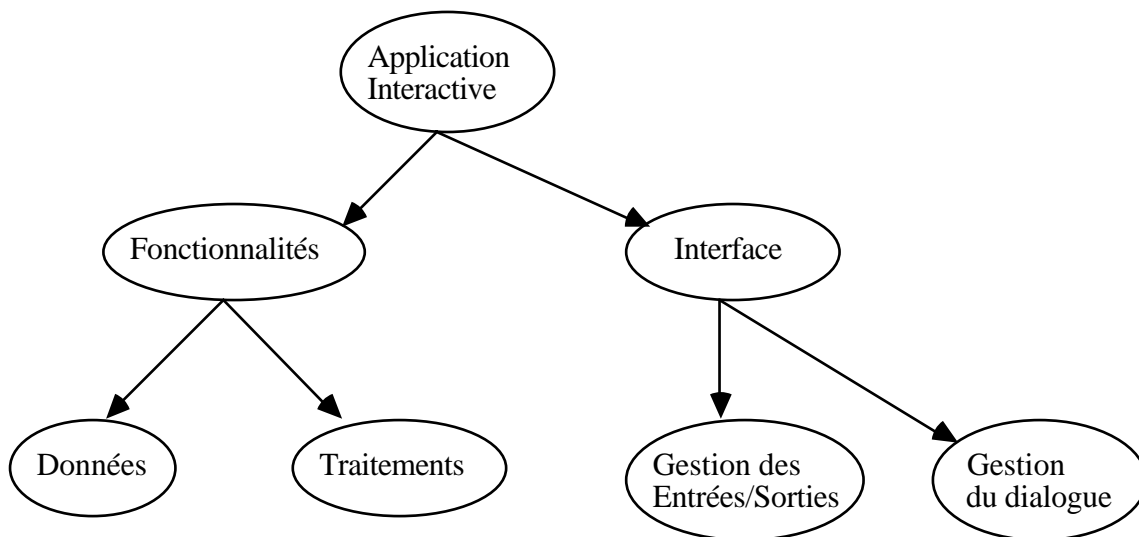
4.1.2 Les caractéristiques d'un SGM pour le programmeur

Nous présentons dans ce paragraphe l'architecture générale d'un SGM puis les propriétés principales et enfin les possibilités de mise en oeuvre (Cou, 86).

4.1.2.1 L'architecture

L'idée essentielle sur l'architecture interne d'un logiciel d'application interactive est de séparer au niveau du code les fonctionnalités concernant l'application de celles concernant l'interface homme/ordinateur.

Cette idée est justifiée par le fait que les fonctionnalités de l'application sont par définition spécifiques à chaque application alors qu'une partie importante des caractéristiques de l'interface est indépendante de toute application (voir les paramètres constants de la Représentation Conceptuelle).



Les fonctionnalités sont décomposées en données et traitements, d'une part pour les facilités ultérieures de modification, d'autre part pour minimiser la programmation en utilisant des systèmes de gestion de base de données.

L'interface est subdivisée en gestion des entrées/sorties, totalement indépendante de l'application, et gestion du dialogue qui fait le lien entre les entrées/sorties et les fonctionnalités de l'application.

L'architecture d'un SGM peut être fermée ou ouverte:

- architecture fermée

Pour cette architecture le SGM est vu comme un tout, fournissant un ensemble bien défini et indissociable de services.

Ce type d'architecture permet de rendre transparents les détails d'implantation et les particularités de la machine, et d'être "insensible" aux erreurs éventuelles des programmeurs d'applications.

L'inconvénient majeur est que cette structure permet difficilement l'évolution vers de nouvelles activités et ne permet pas aux programmeurs d'application d'utiliser de manière particulière les différentes fonctionnalités du SGM.

- architecture ouverte

Cette architecture permet au programmeur d'utiliser tous les niveaux de services dont il a besoin et ainsi d'utiliser des parties du SGM pour les besoins propres de l'activité. Il faut cependant noter qu'une architecture totalement ouverte ne peut garantir un partage équitable des ressources dans un système multiutilisateur.

L'architecture ouverte, pour être opérationnelle, nécessite que :

- chaque module soit effectivement indépendant des autres (pas d'effets de bord, pas de références croisées)

- les services offerts par chaque niveau doivent se limiter au strict nécessaire afin de ne pas nuire aux extensions ultérieures.

Par exemple, la gestion des fenêtres ne gère qu'un niveau de fenêtres (pas les sous-fenêtres) ce qui permet au programmeur d'utiliser ces fonctions de gestion de fenêtres à l'intérieur d'une fenêtre propre à une activité. Alors qu'une gestion de fenêtres plus élaborée gérant dès le départ toute une arborescence de fenêtres n'aurait pas permis une réutilisation facile pour le programmeur.

Toute fonction qui concerne plusieurs niveaux de services doit être définie dans un seul niveau, de priorité plus élevée. Par exemple, le changement d'activité courante.

4.1.2.2 Les propriétés

Le niveau d'abstraction

Il est défini par l'unité d'échange entre l'application et l'interface; cette unité d'échange peut évoluer entre deux extrêmes; elle peut concerner des événements élémentaires comme la prise en compte du "clic" de la souris ou des événements synthétiques comme une commande complète et syntaxiquement correcte.

Ce niveau d'abstraction joue un rôle important pour la facilité de mise en oeuvre de la programmation; en effet, un niveau d'abstraction trop bas correspondant à des événements élémentaires

rend le travail du programmeur lourd et complexe tout en lui donnant le maximum de latitude. L'idéal est bien sûr est d'avoir un niveau d'abstraction élevé tout en se réservant la possibilité de commandes plus élémentaires pour résoudre des cas particuliers.

Contrôle du dialogue

Le contrôle du dialogue correspond aux possibilités d'enchaînement des opérations qui, dans le cas général, sont réparties entre l'utilisateur et la machine; le contrôle spécifie donc les enchaînements automatiques aussi bien que les déclenchements utilisateurs.

Le contrôle peut se faire dans l'application ou dans l'interface.

Il est plus usuel de mettre le contrôle dans l'application et plus précisément dans les traitements, mais cette pratique a l'inconvénient de rendre complexe sinon impossible la modification de ce contrôle par le programmeur et encore plus par l'utilisateur. Aussi, si on veut pouvoir intégrer de nouvelles procédures effectives, il est préférable de mettre ce contrôle dans l'interface.

Par contre, on peut envisager un contrôle réparti entre l'application et l'interface dans le cas où on a un traitement dirigée par les données, c'est-à-dire quand les enchaînements dépendent au niveau du logiciel de la valeur des données; c'est plus précisément le cas dans les systèmes d'aide à la décision et en particulier dans les systèmes experts.

Parallélisme

Nous distinguons le parallélisme géré par l'ordinateur du parallélisme tel qu'il est perçu par l'utilisateur.

Le parallélisme géré par l'ordinateur consiste à faire travailler en parallèle soit plusieurs périphériques, soit plusieurs applications, soit plusieurs utilisateurs.

Ces fonctions sont prise en compte par le système d'exploitation de la machine.

Le parallélisme perçu par l'utilisateur correspond à la possibilité de ce dernier d'interrompre à tout instant une application pour en activer une autre en gardant présent sur l'écran l'image de ces différentes applications. Cette fonction est rendue possible et praticable pour l'utilisateur par la gestion du multifenêtrage au niveau de l'interface, ce multifenêtrage s'accompagnant ou non de multitâche. La fonction de multitâche (ou parallélisme de la machine) a un impact sur le temps de réponse alors que le fonction de multifenêtrage facilite le travail de l'utilisateur.

Contexte

Nous parlons ici du contexte de l'interaction entre l'homme et la machine. La plupart du temps ce contexte n'est pas pris en compte par le logiciel .

On peut distinguer plusieurs formes de contexte:

- le contexte de l'utilisateur qui permet de prendre en compte différents types d'utilisateur
- le contexte des objectifs du travail qui est révélé par la logique d'utilisation
- le contexte de la transaction qui permet de savoir où l'on se situe par rapport aux objectifs recherchés.

La gestion de ces différents contextes a un impact sur le guidage, le traitement des erreurs et l'adaptabilité aux utilisateurs.

Adaptabilité

On peut distinguer l'adaptation gérée automatiquement par l'ordinateur de l'adaptation déclenchée par l'utilisateur. Dans les deux cas, il s'agit de la possibilité de modifier le logiciel pour s'adapter à de nouveaux besoins.

L'adaptation automatique du logiciel est abordée en intelligence artificielle et réalisée dans certains didacticiels.

L'adaptation déclenchée par l'utilisateur n'est possible que si le logiciel est structuré en ce sens; d'où l'importance d'une architecture modulaire des logiciels interactifs telle que nous la présentons ici et qui est une condition nécessaire à toute recherche d'adaptabilité.

4.1.2.3 La mise en oeuvre

L'utilisation d'un SGM implique un style de programmation particulier que l'on qualifie de "dirigée par les événements".

Le programme est conçu comme une boucle appelant régulièrement une procédure qui lui fournit l'événement suivant disponible et qui aiguille le programme vers l'action concernée, en fonction du type d'événement reçu et de l'état courant du programme.

Ce mode de fonctionnement s'apparente aux automates d'états finis où l'état suivant est fonction de l'état courant et de l'événement qui provoque la transition.

Actuellement, les SGM se présentent sous deux formes: les "boîtes à outils" et les systèmes génériques. Les premiers sont disponibles sur le marché alors que les seconds sont en développement.

Les "*boîtes à outils*" sont constituées d'un ensemble de fonctions dédiées à l'interface homme/machine.

Il existe des fonctions à des niveaux d'abstraction très divers; les fonctions élémentaires gèrent des événements comme l'appui sur une touche du clavier ou le cliquage sur la souris alors que d'autres plus synthétiques permettent de gérer les menus.

Les "boîtes à outils" sont des systèmes à architecture ouverte où le programmeur a un maximum de latitude, mais en contrepartie il a à gérer tous les détails de l'interaction ce qui est à la fois complexe et lourd.

Les systèmes génériques fournissent des formes standard à partir desquels on va développer l'application; ils permettent aussi parfois une définition interactive des ressources (dessin d'une fenêtre, d'une icône,...). Ces systèmes sont bâtis sur une boîte à outils; ils peuvent donc avoir une architecture ouverte ou fermée selon que l'on peut ou non accéder aux fonctions de la boîte à outils.

L'objectif de ces systèmes génériques est de faire gagner du temps de programmation quitte à perdre un peu de souplesse. C'est certainement la voie d'avenir pour le plus grand nombre de programmeurs d'application.

4.1.3 Exemples

4.1.3.1 Le SGM de Macintosh

Le SGM de Macintosh est constitué d'une part d'un ensemble de primitives implantées en ROM, d'autre part d'un ensemble de fonctions logicielles regroupées dans la "boîte à outils".

Ces primitives et fonctions sont présentées dans le formalisme PASCAL mais sont également accessibles à partir de tout langage permettant l'exécution de l'instruction TRAP.

Les fonctions de la "boîte à outils" sont regroupées en modules selon le type d'objets qu'elles manipulent; cette structuration correspond à une approche "langage-objet".

Les modules concernant directement l'interface sont:

- le gestionnaire de ressources
- le gestionnaire d'événements
- le gestionnaire de fenêtres
- le gestionnaire de contrôles
- le gestionnaire de menus

D'autres modules sont disponibles pour la programmation de l'application elle-même; ce sont les gestionnaires d'impression, de fichiers, de mémoire.

Le gestionnaire de ressources

Les ressources sont l'ensemble des objets (fenêtres, messages, icônes,...) qui vont être gérés par l'interface; dans ce module on décrit leurs caractéristiques comme, par exemple, la taille d'une fenêtre.

Le fait de décrire ces ressources dans un module séparé permet d'effectuer aisément des modifications.

Il existe des ressources système, accessibles par toute application sans définition préalable.

Le gestionnaire de ressources permet la gestion de l'espace mémoire occupé par ces ressources.

Le gestionnaire d'événements

Ce sont les événements d'interaction avec l'utilisateur, on trouve les événements suivants:

- l'événement "souris" indique que l'utilisateur vient de presser ou de relacher la souris

- l'événement "clavier" indique que l'utilisateur presse ou relache une touche

- l'événement "disque" se produit si l'utilisateur introduit une disquette dans un des lecteurs

- l'événement "d'activation" est généré par le gestionnaire de fenêtres lorsqu'une fenêtre devient active ou inactive

- l'événement de "mise à jour" se produit si une partie du contenu d'une fenêtre doit être dessinée ou redessinée par le programme

- l'événement "réseau" est généré par le gestionnaire du réseau d'ordinateurs quand il existe

- l'événement "privé" peut être défini par n'importe quelle application spécifique

Les événements sont stockés dans une file d'attente gérée en FIFO (premier entrée/premier sortie) mais où on peut définir des priorités.

Le gestionnaire de fenêtres

Il permet de manipuler les fenêtres, c'est-à-dire de les ouvrir, de faire varier leur taille, de les déplacer, de les détruire...

Comme l'utilisateur est libre de superposer ces fenêtres, c'est le gestionnaire de fenêtres qui va gérer cette superposition.

Il permet également de donner des réponses standard à certaines actions de l'utilisateur comme:

- rendre active une fenêtre après "cliquage" de la souris dans cette fenêtre

- fermer une fenêtre après "cliquage" dans l'icône de fermeture

- déplacer une fenêtre après "cliquage" sur la barre de titre.

Il existe des fenêtres prédéfinies mais il est possible de définir de nouveaux types de fenêtres.

Le gestionnaire de contrôles

Les contrôles sont des objets graphiques prédéfinis (boîtes, boutons, barres de défilement) qui permettent de proposer des choix à l'utilisateur. On peut les utiliser, par exemple, pour définir les paramètres d'une impression. Les barres de défilement permettent à l'utilisateur de faire glisser le contenu d'une fenêtre de manière à visualiser plus d'informations que ne peut en contenir une fenêtre. L'utilisateur agit sur ces contrôles par l'intermédiaire de la souris.

Le gestionnaire de menus

Il permet la manipulation de "menus déroulants" qui apparaissent sur l'écran uniquement quand l'utilisateur appuie sur la souris. Il permet également d'activer ou de désactiver des commandes du menu ce qui est utile pour visualiser un menu dynamique.

Nous pouvons constater que beaucoup de ces fonctions présentées dans le formalisme PASCAL sont de bas niveau et demandent donc un travail important de programmation.

Par contre, il existe en LELISP un certain nombre de fonctions de haut niveau qui permettent une mise en oeuvre rapide qu'il peut être intéressant d'utiliser pour construire un prototype d'interface.

4.1.3.2 Les SGM sous UNIX

Nous présentons ici des SGM développés sous UNIX de manière générale et d'un système particulier également sous UNIX : BRUWIN (Mer, 81).

Les SGM. développés sous UNIX offrent à l'utilisateur une commande explicite de création de fenêtre. Celle ci associe automatiquement un interpréteur de commande (shell) à la fenêtre créée, ce qui garantit l'indépendance des programmes qui s'exécutent dans les différentes fenêtres.

La compatibilité des SGM avec UNIX est assurée de la manière suivante :

- le gestionnaire de fenêtre doit collaborer avec le gestionnaire de fichiers. En effet, dans UNIX, toutes les entrées/sorties se font au moyen de fichiers spéciaux. Les programmes d'application accèdent à leur fenêtre par leur fichier standard d'entrée ou de sortie ; ainsi chaque fenêtre simule un clavier : écran alphanumérique (il faut créer des procédures spéciales pour pouvoir dessiner dans une fenêtre);

- ces différents fichiers spéciaux sont gérés à l'aide d'appels systèmes qui sont également utilisés pour gérer les fenêtres puisqu'elles sont considérées comme des fichiers spéciaux;

- la description du contenu de la fenêtre peut être gérée entièrement par le SGM ou bien déléguée au programmeur d'application. Dans ce cas, le SGM a une routine d'interruption qui permet d'activer le programme qui s'occupe de l'affichage du contenu;

- les SGM sont intégrés au moyen d'UNIX car c'est la seule méthode qui permet de partager, entre plusieurs programmes à la fois, du code et des données. Certains opérateurs offerts par les systèmes UNIX font double emploi avec un SGM:

- . l'opérateur postfixe "&" qui permet d'activer une commande sans attendre que celle qui est active soit terminée
- . l'opérateur préfixe "!" qui permet d'envoyer directement une commande à l'interpréteur de commandes sans quitter son programme.

Exemple de SGM sous UNIX : BRUWIN

Le système BRUWIN (Brown University Window Manager) est un des premiers SGM réalisés sous UNIX. Il fonctionne avec des terminaux alphanumériques permettant de désigner un point sur l'écran.

Il offre deux modes d'utilisation :

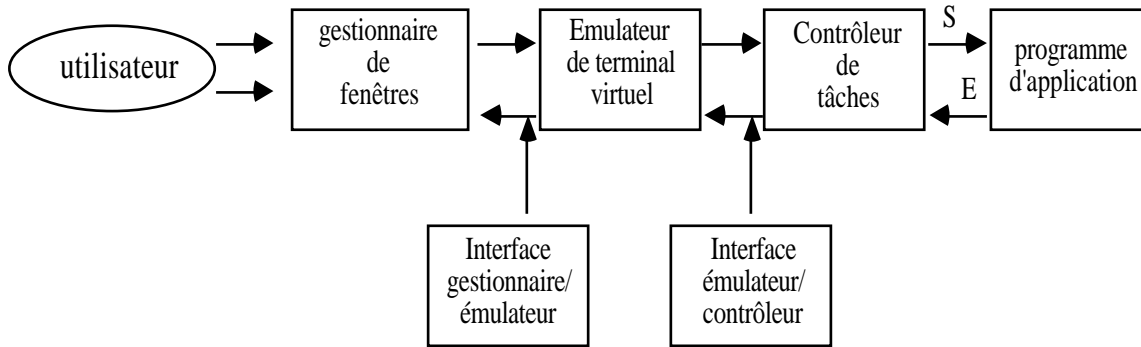
- le mode processus pour communiquer avec le programme en exécution dans la fenêtre courante
- le mode commande pour communiquer avec le gestionnaire de fenêtres.

Ce mode offre les commandes suivantes :

- . cancel : supprime l'exécution de la commande BRUWIN en cours
- . create : crée une fenêtre (associée à l'interpréteur de commande)
- . change : modifie la taille d'une fenêtre
- . move : déplace une fenêtre
- . destroy : détruit une fenêtre et son interpréteur de commandes
- . command : change de fenêtre courante.

BRUWIN est divisé en trois modules principaux :

- . un module gestionnaire de fenêtre qui est responsable de la division de l'écran en fenêtres, de la mise à jour de l'écran et de la gestion des commandes de manipulation de fenêtres;
- un module émulateur de terminal virtuel qui permet l'adaptation à différents types de terminaux.
- . un module contrôleur de tâches qui assure le lien entre BRUWIN et le système d'exploitation. Il est également responsable de l'envoi des caractères au programme d'application courant.



L'indépendance entre ces trois modules est assurée par deux modules interface (gestionnaire / émulateur et émulateur / contrôleur) ; ce qui permet de modifier l'un d'entre eux sans modifier l'ensemble.

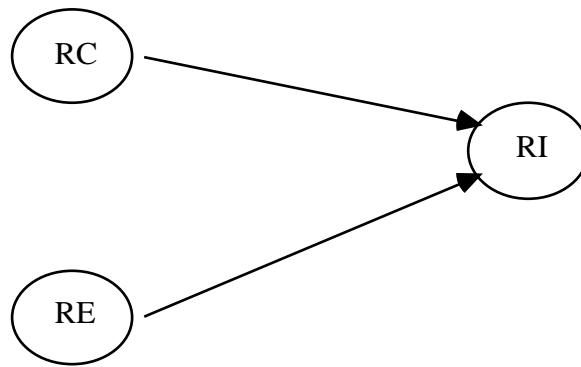
Cet exemple montre que le SGM, tel que nous l'avons défini, peut être vu comme une extension du système d'exploitation, UNIX se prêtant bien à ce genre d'extension.

4.2 Traduction des Représentations Conceptuelle et Externe

La Représentation Interne n'est traitée ici que très succinctement par le biais de son architecture générale, car une présentation détaillée dépasserait par le thème et le volume le cadre de ce livre :

- par le thème, car il serait nécessaire de rentrer dans le détail des structures de contrôle comme les Réseaux de Petri et dans le détail des techniques de programmation alors que ce livre est centré sur l'analyse et la spécification des applications interactives;
- par le volume, car un tel développement nécessite à lui seul un livre.

La R.I. est la traduction des Représentations Conceptuelle et Externe. Normalement toutes les spécifications ont déjà été faites et le passage à la R.I. est une simple traduction sans génération de nouvelles spécifications.

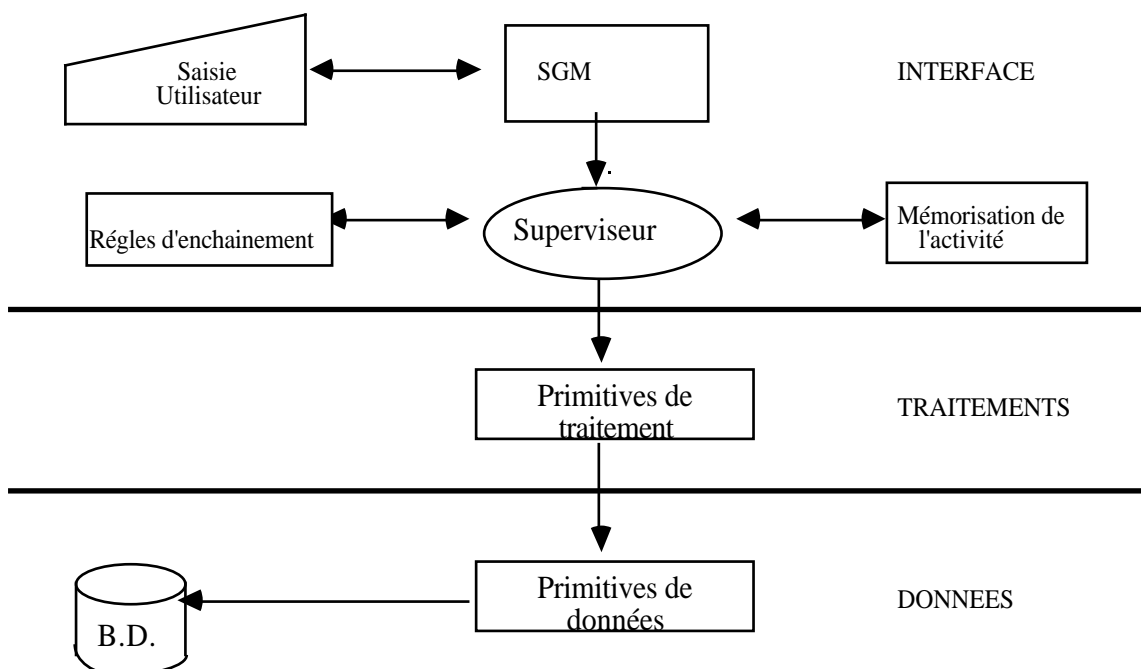


4.2.1 Architecture générale de la Représentation Interne

Notre objectif principal est de permettre la conception et la réalisation d'interfaces le plus adaptative possibles. Cette préoccupation se traduit dans la conception de l'architecture générale de la R.I.

Les modifications de programmes sont d'autant plus aisées que les différentes fonctionnalités sont séparées de telle manière que la modification d'un élément n'ait pas de répercussions sur les autres éléments (sauf au niveau de l'interfaçage des différentes fonctionnalités).

Nous représentons cette architecture générale sur le schéma suivant :



Nous définissons une primitive comme un élément indécomposable vis-à-vis du système informatique, de même qu'une opération élémentaire est un élément indécomposable vis-à-vis de l'utilisateur.

Les primitives de traitement décrivent les algorithmes de calcul.
Les primitives de données décrivent les contrôles de vraisemblance, de concordance, la recherche et la mise à jour des données.

Les règles d'enchaînement décrivent les différentes procédures (prévues, minimales et effectives).
La mémorisation de l'activité des utilisateurs permet de gérer les interruptions, les différés, les aides et l'état d'avancement de la transaction en cours.

Le superviseur joue un rôle central d'activation des différents modules en fonction du contexte; il a un double rôle d'enchaînement des opérations et de contrôle de la validité de ces enchaînements.

Le SGM détermine la nature de l'événement reçu (clavier, lecteur,...) et le traite s'il relève seulement de la syntaxe (déplacement fenêtre, changement de taille,...); s'il relève de la sémantique de l'application (commandes,...), il active le superviseur.

Les différents modules que nous venons de définir n'ont pas le même statut; certains ont un caractère général alors que d'autres sont entièrement définis pour chaque application. Nous présentons ces différents modules en précisant leurs rôles et leurs structures.

Le superviseur

C'est le module qui a le caractère le plus général, c'est-à-dire qu'il est totalement indépendant de tout type d'application.

Le superviseur a deux fonctions imbriquées :

- une fonction d'enchaînement,
- une fonction de contrôle.

Il reçoit les demandes d'actions de l'utilisateur à travers le SGM.

Il contrôle le fait que la demande de l'utilisateur est compatible avec les règles d'enchaînement et la mémorisation de l'activité.

Si tel est le cas, il active les traitements demandés et il met à jour la mémorisation de l'activité.

Il renvoie des messages à l'utilisateur par le biais du SGM.

De plus il gère les abandons, les reports, les interruptions (dans le cas où cette dernière fonction n'est pas prise en compte par le SGM) par des mises à jour du module de mémorisation de l'activité.

Règles d'enchaînement et mémorisation de l'activité

Pour ces modules nous avons deux possibilités de représentation: les automates d'états finis et les Réseaux de Petri.

La représentation sous forme d'automates d'états finis est maintenant classique et bien connue des informaticiens; la représentation visuelle est simple et la transposition sous forme vectorielle facile. Le seul inconvénient est qu'elle ne permet pas une gestion aisée du parallélisme.

Le parallélisme étant une composante essentielle des traitements en informatique des organisations et dans les interfaces homme/machine, il est plus exact d'utiliser le formalisme des Réseaux de Petri (Bra, 83) et plus précisément les Réseaux de Petri à Objets (Sib, 85) et (Bar, 86) qui ont l'avantage de permettre l'expression des interactions entre les données et les traitements.

Les primitives de traitement et de données

Nous incluons dans ces modules les primitives de :

- contrôle de données (vraisemblance et concordance)
- gestion de données
- gestion de traitements.

La définition de ces primitives est entièrement particulière à chaque application.

Elles sont définies pour chaque opérations interactive ou automatique.

L'architecture que nous venons de présenter correspond aux possibilités maximales de mise en oeuvre des fonctionnalités des logiciels interactifs; il n'est pas évident de réunir l'ensemble des logiciels et des compétences permettant ce type de mise en oeuvre; nous examinons ce qui peut être fait si certains des composants sont absents:

- si on ne dispose pas d'un interpréteur de Réseau de Petri, on peut gérer les enchaînements et la mémorisation de la transaction au moyen de vecteurs booléens qui traduisent le comportement de l'automate d'état fini; ceci sera praticable tant que le parallélisme est faible;

- si on ne dispose pas d'un SGM, certaines fonctionnalités comme la gestion des interruptions peuvent être prises en charge par le superviseur ou éventuellement par une modification du système d'exploitation; par contre, la plupart des fonctionnalités offertes directement à l'utilisateur comme le déplacement de fenêtres ou leur superposition ne peuvent plus raisonnablement être gérées;

- si on adopte une architecture de programmes plus classique où l'on ne distingue pas l'interface et les traitements, on se prive de possibilités de modification et en particulier d'adaptation après des tests auprès des utilisateurs; mais on peut quand même mettre en oeuvre les recommandations ergonomiques à caractère général.

Chapitre 5 :

DIANE

METHODE DE CONCEPTION D'APPLICATIONS INTERACTIVES

Comme nous l'avons défini au chapitre 1, une méthode permet de rendre opérationnel un modèle ou en d'autres termes constitue un mode d'emploi particulier d'un modèle. En ce sens, elle restreint les possibilités du modèle et de fait, quand on a très bien intégré un modèle, on n'a plus besoin de suivre une méthode particulière et on s'adapte mieux ainsi aux particularités de l'environnement.

Par contre, il est beaucoup plus simple d'apprendre à manier un nouveau modèle par le biais d'une méthode qui sert de guide tout au long de la démarche.

On peut dire qu'une méthode est une sorte de procédure prévue qui indique:

- comment s'articulent les différents éléments du modèle (séquentialité, parallélisme),
- quels sont ceux qui sont observables et comment,
- quels sont ceux qui sont déductibles et à partir de quoi (observations, connaissances générales).

On peut ainsi à la limite appliquer une méthode sans maîtriser pleinement le modèle sous-jacent. A cet effet, ce chapitre constitue un tout en soi qui peut être utilisé quasiment indépendamment des autres chapitres. Mais, si on veut savoir sur quels éléments de psychologie cognitive et d'ergonomie se base cette méthode, il faut lire la première partie des chapitres 2 et 3.

Pour les définitions de concepts, on se reportera au glossaire (les concepts définis dans le glossaire et utilisés pour la première fois dans ce chapitre sont marqués d'une astérisque "*").

Le chapitre comporte deux parties:

- dans les trois premiers paragraphes, nous présentons une démarche informatique classique où l'informaticien est responsable de l'ensemble du processus de l'analyse jusqu'à la conception de l'application. Nous présentons d'abord l'étude de l'existant, puis la conception de l'application automatisée et en dernier lieu les tests sur le prototype ainsi défini;

- dans le dernier paragraphe, nous traitons une étude de cas complète qui va de l'étude de l'existant à la définition de la Représentation Externe.

La méthode DIANE que nous exposons ci-dessous concerne exclusivement l'analyse et la conception des applications interactives ; mais la plupart du temps, il est nécessaire de faire une étude globale du système d'information avant de se préoccuper des applications interactives, il nous a paru donc important de montrer comment la méthode DIANE pouvait se greffer sur une méthode d'analyse générale. Dans la suite de cet exposé, nous montrons comment intégrer la méthode DIANE à partir de la méthode MERISE.

5.1 Etude de l'existant

Nous rappelons dans le premier paragraphe les divers modes de recueil de l'information d'un système existant afin de pouvoir s'y référer dans le deuxième paragraphe quand nous décrivons le système existant, l'objectif étant de montrer quels sont les modes de recueil les plus adaptés pour décrire les différents paramètres du système.

5.1.1 Collecte de l'information

Nous présentons succinctement l'ensemble des techniques utilisables pour décrire le travail existant; dans les trois premières (entretien, réunion, questionnaire), l'utilisateur joue un rôle actif d'explicitation verbale de son travail alors que dans les trois dernières (autorelevé, observation, capteur), il exécute son travail sans l'analyser.

Entretien

C'est certainement la technique la plus utilisée par les informaticiens ; elle présente l'avantage d'être une très bonne source d'informations qualitatives.

L'entretien peut être structuré ou non; s'il n'est pas structuré, il permet d'obtenir des informations à caractère général, de percevoir le climat et les problèmes principaux qui se posent. Un entretien non structuré constitue souvent une étape obligatoire pour commencer une étude de l'existant dans une entreprise peu ou pas connue.

L'entretien structuré s'apparente à un questionnaire en ce sens qu'il demande une préparation préalable de questions à réponses limitées

ou ouvertes; s'il est bien conçu, il facilite la synthèse des informations provenant des différents interlocuteurs.

Le choix des personnes à interroger est un problème essentiel, car il est rarement possible d'avoir des entretiens avec l'ensemble des personnes intéressées par l'étude et un mauvais choix peut donner une vision fautive ou biaisée de l'existant.

L'inconvénient majeur de l'entretien est qu'il est coûteux en temps, à la fois pour l'analyste et pour les utilisateurs . Aussi peut-on le remplacer ou le compléter par des réunions ou des questionnaires.

Réunion

La conduite de réunion suppose un minimum de formation et d'expérience pour être efficace ainsi qu'un choix judicieux du nombre et du type des participants. Car les interactions entre les participants peuvent être stérilisantes et empêcher de recueillir toutes les informations souhaitables.

Par contre, correctement mené elle peut permettre une adhésion du groupe à l'étude et l'émergence de nouvelles propositions.

Questionnaire

Les questionnaires remplis par l'utilisateur peuvent compléter ou même remplacer des entretiens ou des réunions pour recueillir des informations précises, bien délimités et quantitatives.

L'inconvénient réside dans le fait qu'il peut être mal rempli si l'utilisateur interprète de manière erronée certaines questions.

Les entretiens, réunions et questionnaires font appel explicitement à l'utilisateur et donc à sa subjectivité pour décrire sa situation de travail; on obtient donc ainsi le réel perçu par l'utilisateur et exprimé pour l'analyste.

Cette situation peut avoir trois effets:

- certains comportements inconscients ne sont pas exprimés,
- si le réel est complexe, on recueille la description des cas "normaux" et non pas des cas effectifs,
- certains faits peuvent être volontairement omis.

Autorelevé

Il consiste à faire remplir aux intéressés un questionnaire détaillée décrivant leur travail au fur et à mesure qu'il se déroule; on recueille ainsi le travail effectif et non pas le travail décrit.

L'inconvénient majeur est que l'analyste peut se retrouver avec une masse d'information inexploitable qui ne lui permet pas de dégager une synthèse du travail ainsi décrit.

Observation

Cette technique très utilisée par les ergonomes l'est fort peu par les informaticiens.

Elle permet, comme l'autorelevé, de recueillir une information sur le travail effectif ; son avantage supplémentaire est de ne pas mobiliser l'utilisateur puisque c'est un observateur extérieur qui remplit les relevés.

Elle est par contre très coûteuse en temps d'analyste.

Capteur

Dans le cas d'étude sur des logiciels existants, cette technique consiste à rajouter des éléments aux logiciels de manière à recueillir automatiquement les traces de travail de l'utilisateur. C'est une observation automatique qui de ce fait ne s'applique qu'à l'observation de certains paramètres, mais qui peut compléter d'autres techniques d'observation.

Les autorelevés, les observations et les capteurs permettent par des moyens différents de recueillir les traces de travail effectif.

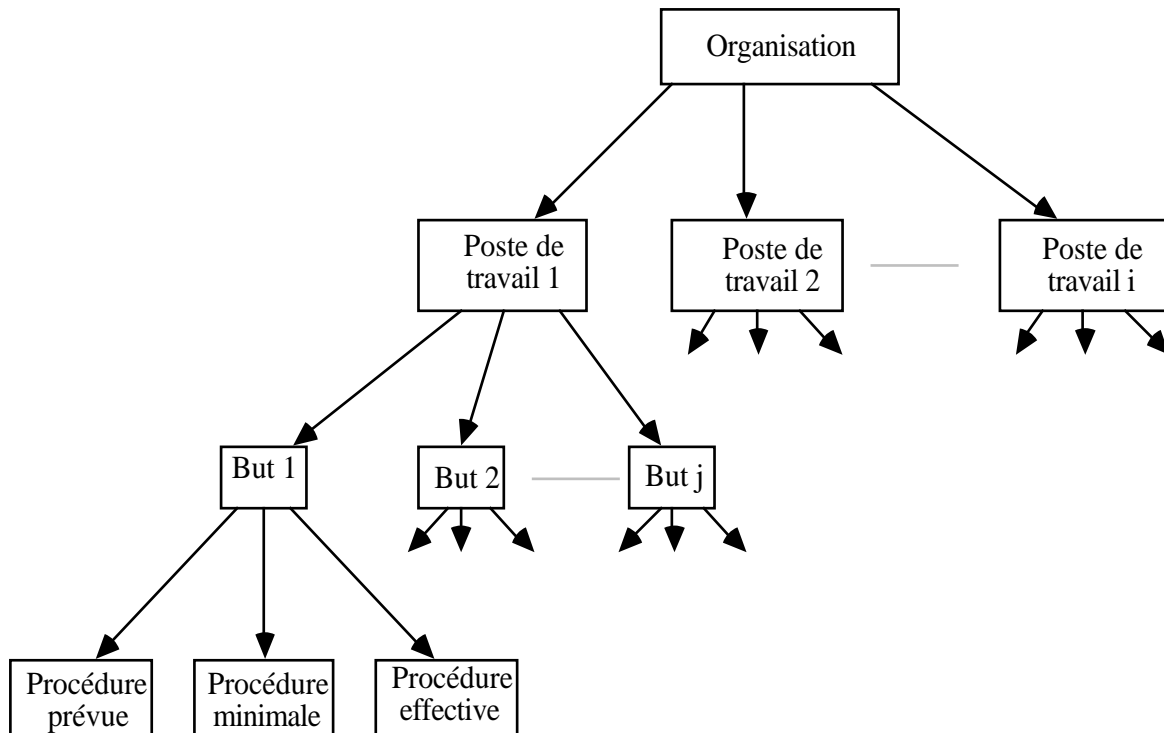
La plupart du temps, on combine plusieurs de ces techniques pour avoir l'image la plus exacte du système existant; on peut, par exemple, procéder à des observations, puis avoir un entretien avec l'utilisateur observé pour obtenir des compléments d'informations sur les faits observés.

Quel que soit le soin que l'on prend à recueillir l'information sur le système existant, on n'obtient qu'une image partielle du système réel et l'essentiel est de savoir quelle partie du système réel on a décidé d'ignorer, jusqu'à quel niveau de détail on veut le décrire et quels sont les éléments du système réel que l'on juge négligeables.

5.1.2 Description du système existant

L'objectif de cette étape est de montrer comment déterminer les logiques d'utilisation (*) associées aux différents postes de travail concernés par l'automatisation.

Cette logique d'utilisation se traduit ici par la détermination des buts (*) d'un poste de travail (*); puis, pour chaque but, la détermination des différentes procédures (*) permettant de l'atteindre.



5.1.2.1 Les buts d'un poste de travail

Le découpage d'un système d'information se fait usuellement dans les méthodes d'analyse par la notion de domaine ou de fonction. L'objectif est de découper le système d'information en sous-systèmes le plus indépendants possibles (c'est-à-dire ayant le minimum de relations entre eux).

Dans la méthode MERISE, à l'intérieur d'un domaine ou d'une fonction du système d'information, le découpage, s'il y a lieu, se fait en processus au niveau conceptuel et en procédures au niveau organisationnel. Le processus ou la procédure est elle-même décomposée en un enchaînement d'événements et d'opérations.

L'opération est définie "comme un ensemble d'actions élémentaires non ininterrompible".

La notion d'opération non interrompible est une notion importante du découpage de MERISE qui ne pose aucun problème pour les opérations manuelles et automatiques, mais qui nous semble devoir être précisée dans le cas d'opérations interactives ; aussi définissons-nous une opération interactive comme "*un ensemble de transactions qui peuvent être exécutées consécutivement et sans attente d'événements externes au système homme-ordinateur*". Cette définition permet d'intégrer toutes les interruptions provoquées par l'utilisateur en cours d'opération que ce soit pour activer d'autres opérations ou pour différer son travail.

Rechercher les buts d'un poste de travail revient à chercher **la logique d'utilisation** des utilisateurs. Pour cela, nous pouvons procéder de deux manières :

- la première manière part du système d'information
- la seconde manière est issue de l'étude du système de décision.

Détermination des buts à partir du système d'information

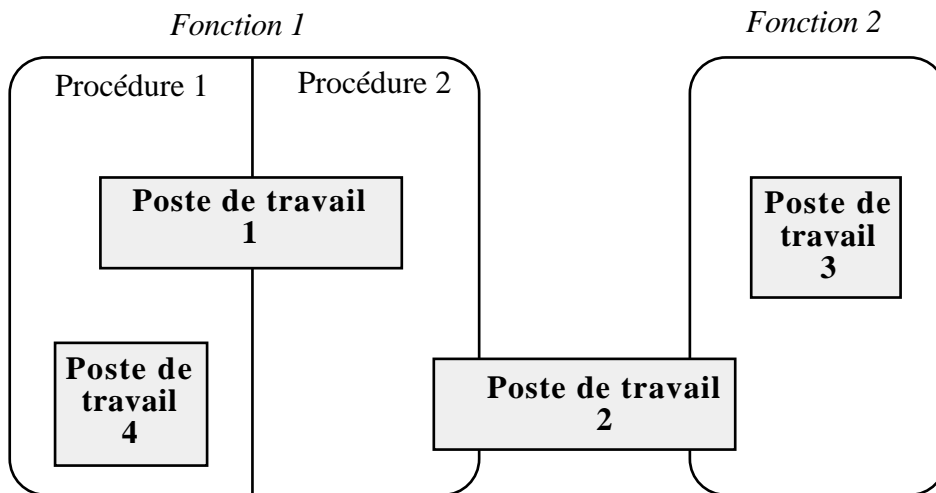
Nous présentons en premier la méthode issue de l'étude du système d'information, car c'est celle qui est la plus usuelle pour les analystes. En effet les méthodes de conception des systèmes informatisés s'appuient sur le système d'information et même, souvent, uniquement sur le système d'information du système opérant.

Dans la méthode MERISE, la détermination des buts d'un poste de travail se situera immédiatement après la description du niveau organisationnel des traitements et des données du système existant. A la fin de cette phase, nous disposons, pour les traitements, d'un ensemble de procédures qui décrivent les enchaînements des opérations du nouveau système et pour chaque opération nous avons déterminé, sa nature (interactive, automatique, manuelle) et son lieu d'exécution (poste de travail, service). Pour les données, nous disposons de l'ensemble des vues externes du poste de travail. Par la suite, nous considérons simultanément les traitements et les données d'une opération.

Nous allons maintenant regrouper toutes les opérations interactives et automatiques exécutées sur un même type de poste de travail. Il faut remarquer que le même type de poste de travail peut participer à des **procédures** différentes mais aussi à des **fonctions différentes**.

(Ex : la gestion des stocks participe à la fonction achats et à la fonction ventes ; c'est le seul point d'intersection de ces fonctions). Ces regroupements seront faits en respectant les règles d'enchaînement et les synchronisations (*) déterminées aux niveaux précédents (règles d'enchaînement du niveau conceptuel **et** du niveau organisationnel).

Exemple d'interaction entre fonctions, procédures et opérations d'un poste de travail:



Pour faire émerger les buts , nous remarquons qu'un but d'un poste de travail donné correspond à un sous-ensemble d'opérations de l'ensemble des opérations possibles de ce poste de travail.

Il est important de noter que l'ensemble des buts d'un poste de travail *ne constitue pas forcément une partition de l'ensemble des opérations* car une même opération peut appartenir à deux buts différents.

Pour déterminer si deux opérations appartiennent au même but (c'est-à-dire sont nécessaires à la réalisation du but), il faut demander à **l'utilisateur** s'il peut avoir besoin en même temps de ces opérations dans la réalisation d'une de ces tâches. L'objectif est qu'il ait "sous la main", à un moment donné, l'ensemble des opérations qu'il peut être amené à utiliser pour réaliser un but donné. Il faut penser en particulier aux opérations de consultation pour **l'aide** et aux opérations de modification et d'annulation pour la **correction des erreurs**.

Les informations nécessaires aux regroupements des opérations en buts peuvent être recueillies par entretien auprès des utilisateurs concernés (exécutants); en cas d'ambiguïté, les entretiens peuvent être complétés par des autorelevés ou des observations.

Cette démarche, qui part des opérations du système d'information pour faire émerger les buts, convient bien aux postes de faible responsabilité qui n'ont pas de fonction de pilotage vis-à-vis d'autres

postes de travail et/ou qui n'ont pas de fonction de régulation importante.

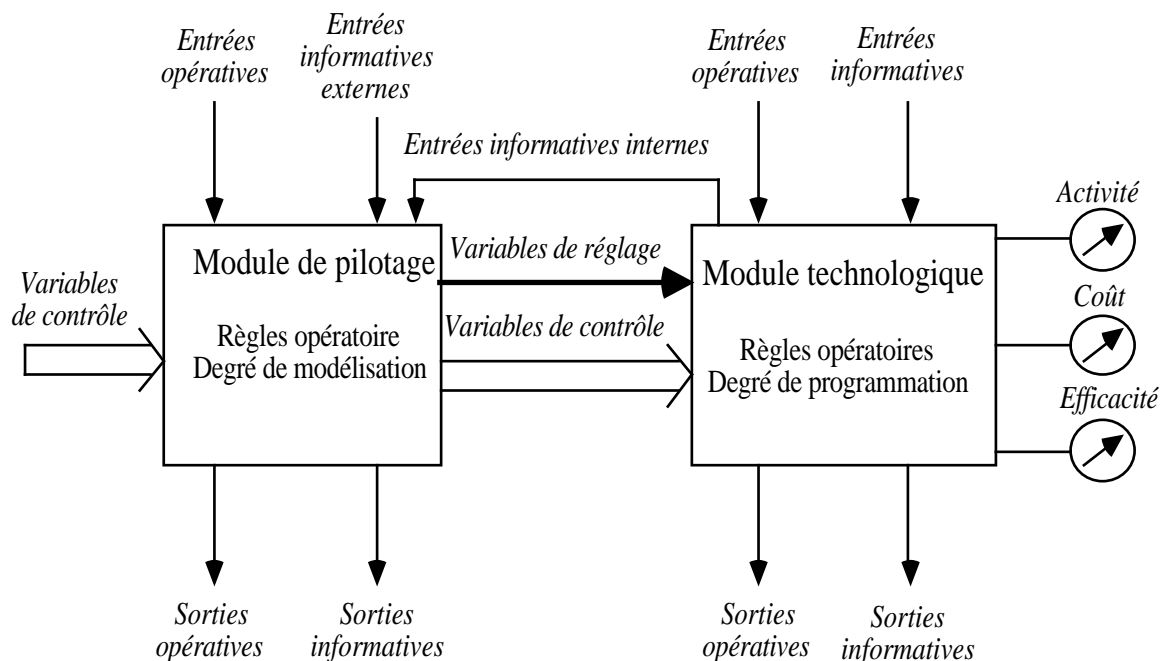
Dans les autres cas, il est plus efficace et plus sûr d'utiliser la seconde démarche qui part de l'étude du système de décision.

Détermination des buts à partir du système de décision

Le système de décision est peu étudié en tant que tel dans les méthodes d'analyse informatique; la seule méthode opérationnelle qui l'étudie ainsi que ses interactions avec le système d'information est l'Analyse Modulaire des Systèmes de J. Melese (Mel,77). Cette méthode, que nous rappelons brièvement ci-dessous, peut être appliquée globalement ou localement; elle permet de dégager les objectifs d'un poste de travail et d'étudier la compatibilité des boucles de régulation par rapport à ces objectifs. Elle peut être un guide pour dégager les objectifs mais aussi pour définir les différents types de procédures, en particulier la procédure minimale.

L'Analyse Modulaire des systèmes

Le principal intérêt de l'A.M.S est de permettre l'analyse de l'interaction entre le système d'information et le système de décision de l'entreprise. Pour cela, elle propose une description où toute activité de l'entreprise est modélisée sous la forme d'un couple module de pilotage-module technologique que nous représentons dans un formalisme simplifié sur le schéma suivant:



Le module de pilotage permet de décrire les mécanismes de décision qui transforment les informations d'entrée de ce module en informations de décision pour le module technologique afin de maintenir le fonctionnement du module technologique en direction des objectifs assignés. Les objectifs sont décrits selon trois types de variables qui sont l'activité, le coût, l'efficacité. Pour notre étude, ce sont les descriptions des objectifs en termes d'activité et d'efficacité qui sont à retenir.

Un autre concept intéressant pour notre étude est la distinction entre variables de réglage et variables de contrôle. Les variables de contrôle correspondent à une transmission de décisions prises à un niveau hiérarchique supérieur; le module de pilotage se contente d'adapter ces décisions au contexte du module technologique, mais il n'en est pas à l'origine et il ne peut les remettre en cause.

Les variables de réglage sont par contre de l'entière responsabilité du module de pilotage correspondant. Elles permettent de décrire les décisions qui sont générées à ce niveau en fonction des informations provenant du module technologique (entrées informatives internes) ou provenant d'autres modules ou de l'environnement (entrées opératives et informatives externes).

L'ensemble des variables de réglage et des entrées d'information constitue une boucle de régulation qui a pour but de maîtriser le fonctionnement du module technologique et d'absorber une partie des fluctuations de ce module.

Cette décomposition en deux modules s'applique aussi bien au niveau de l'entreprise globale qu'au niveau du poste de travail. Elle permet ainsi de dégager des chaînes de pilotage qui décrivent le système de décision de l'entreprise, c'est-à-dire la transmission des décisions, la répartition des responsabilités et les boucles de régulation.

Dans le cadre de notre étude, nous l'appliquons à l'ensemble des postes de travail concernés par le nouveau système afin de dégager à la fois les buts et la latitude décisionnelle de chaque poste de travail. La latitude décisionnelle permet de dégager plus aisément les procédures minimales.

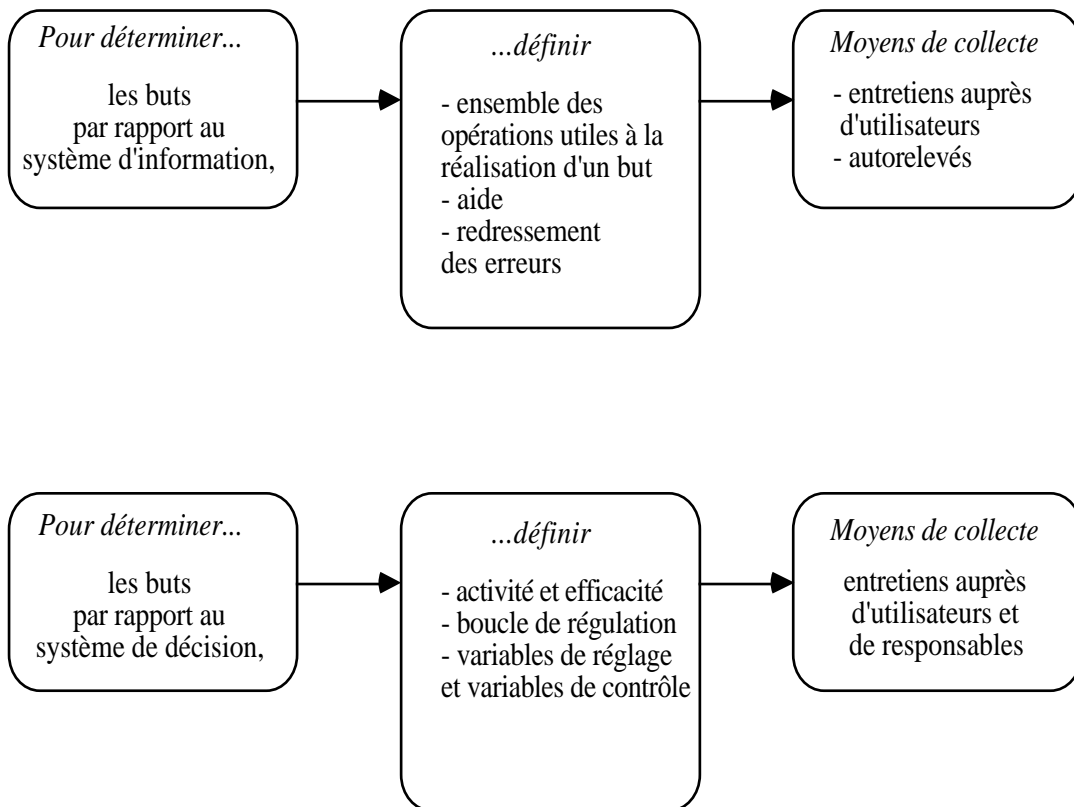
Le reste du système d'information est décrit au niveau du module technologique par les règles opératoires permettant de transformer les informations d'entrée de ce module en informations de sortie.

Les objectifs ou buts étant ainsi déterminés par l'étude du système de décision et l'ensemble des opérations par l'étude du système d'information, il suffit de mettre les deux en correspondance en se

demandant quelles sont les opérations qui contribuent à la réalisation d'un but.

Cette démarche est sans contexte plus sûre que la précédente, mais comme elle est plus lourde à mettre en oeuvre il est préférable de la réserver aux postes exerçant des responsabilités dans les organisations ou aux postes dont on n'arrive pas à faire émerger sans ambiguïté des buts par le simple regroupement des opérations.

Tableau de synthèse de la détermination des buts:



5.1.2.2 Les différentes procédures existantes

Les procédures, que nous définissons ici, utilisent un formalisme, présenté dans le chapitre 2, très voisin de celui utilisé dans MERISE pour décrire les processus ou procédures; mais *elles se distinguent des procédures de MERISE par leur sémantique car elles décrivent l'enchaînement des opérations nécessaires à la réalisation d'un but d'un utilisateur et non pas un sous-domaine du système d'information.*

La notion d'opération (*) est elle aussi modifiée pour pouvoir intégrer les interruptions à l'initiative de l'utilisateur.

Pour chaque but, nous déterminons *en premier la procédure prévue* (*) car c'est celle qui se dégage "naturellement" d'une étude de l'existant menée au moyen d'entretiens ou de questionnaires auprès des exécutants ou des responsables.

En effet, quand on interroge des utilisateurs, on obtient en premier lieu la description des cas les plus courants qui correspondent normalement à la procédure prévue. La description de la procédure prévue existante est indispensable pour la suite de l'étude.

Pour obtenir la description de *procédures effectives* (*) qui correspondent à des cas particuliers ou à des habitudes de travail différentes, il faut soit mener des entretiens plus approfondis auprès des exécutants, soit procéder à des observations ou mettre en place des capteurs ou faire remplir des autorelevés. La description des procédures effectives existantes étant très coûteuse en temps d'analyse, elle ne doit être faite que si les procédures existantes sont peu ou pas modifiées par le changement de système.

La *procédure minimale* (*) ne peut se dégager spontanément des entretiens car elle comprend des informations qui, la plupart du temps, ne sont pas explicites. Il s'agit ici de définir explicitement la latitude décisionnelle de l'utilisateur ou, en d'autres termes, la marge de manoeuvre dont il dispose pour effectuer le travail assigné à son poste. Cette marge de manoeuvre s'exprime par la nature des déclenchements (*) (optionnel ou systématique), des précédences (*) (permanentes ou indicatives) et des opérations (*) (obligatoires ou facultatives).

Si le système existant que nous observons est composé de procédures entièrement manuelles, la nature des déclenchements, des précédences et des opérations décrira la répartition des responsabilités entre le poste de travail que nous décrivons et les postes de travail dont il dépend ou qui dépendent de lui.

Tous les aspects obligatoires, permanents et systématiques correspondent à des consignes intransgressables à ce niveau de responsabilité.

Si les procédures existantes sont déjà automatisées, ces consignes intransgressables doivent être déjà intégrées aux logiciels; en conséquence, il faut dans ce cas étudier la répartition du pilotage entre l'homme et la machine. Mais la répartition existante du pilotage entre l'homme et la machine ne correspond pas forcément à la procédure minimale, nous pouvons simplement supposer que la procédure minimale est incluse dans la partie de la procédure gérée par ordinateur et qu'il faut alors la dégager par une étude plus approfondie.

La procédure minimale ne peut être recueillie que par entretien dirigé auprès des responsables ou des exécutants selon le contexte de l'entreprise.

Pour déterminer la procédure minimale, le plus simple est de partir de la procédure prévue déjà recueillie et de poser des questions permettant de dégager les déclenchements systématiques, les précédences permanentes et les opérations obligatoires.

La notion de déclenchement n'a de pertinence que dans le cas d'une procédure existante déjà automatisée (opérations interactives ou automatiques) où l'on se demande si c'est l'utilisateur ou l'ordinateur qui déclenche une opération; en effet, si les opérations sont manuelles, il est évident qu'elles ne peuvent être déclenchées que par l'utilisateur!

Nous rappelons qu'il y a des précédences de type logique (une modification ne peut avoir lieu qu'après une création) et des précédences de type organisationnel, c'est-à-dire liées aux règles de gestion ("tout paiement de facture ne peut se faire qu'après l'émission d'un bon de commande").

Ce sont les précédences de type organisationnel que nous étudions ici.

Pour trouver la nature des précédences, on pose des questions du type:

"Que se passerait-il si vous effectuiez l'opération Y avant l'opération X ?" (quand la procédure prévue indique que X est exécutée avant Y).

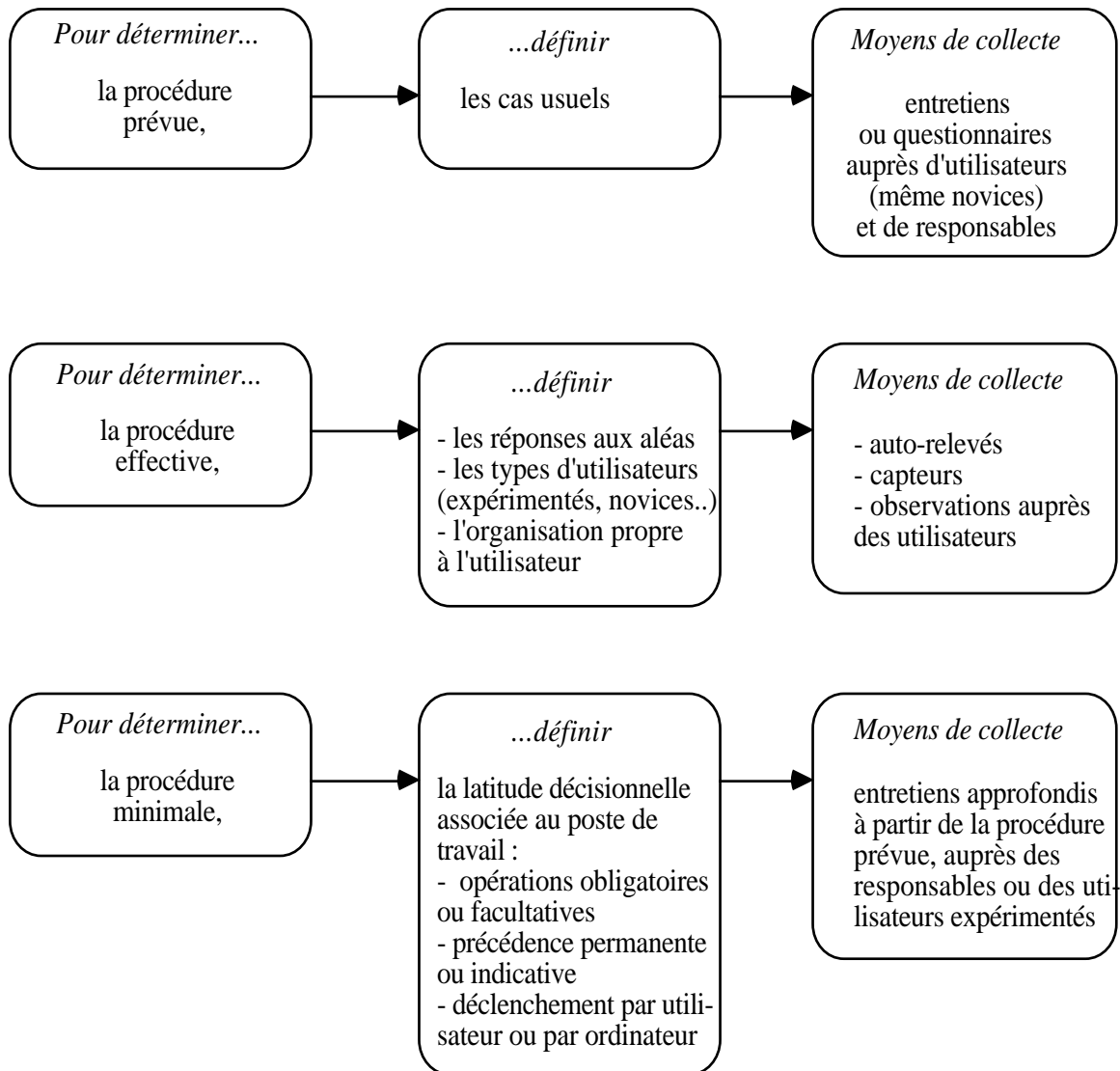
Si l'utilisateur répond que c'est impossible parce que le règlement interne ou la loi s'y oppose ou toute autre raison ne relevant pas de lui, on a une précedence permanente. S'il répond que c'est plus commode ou plus logique ou plus facile, on a très certainement affaire à une précedence indicative.

Pour connaître le caractère obligatoire ou facultatif des opérations mises en oeuvre dans la procédure prévue, il faut poser des questions du type

"Peut-on ne pas exécuter l'opération X ?" ou

"Que se passe-t-il si on ne fait pas l'opération X ?".

Tableau de synthèse de la détermination des procédures:

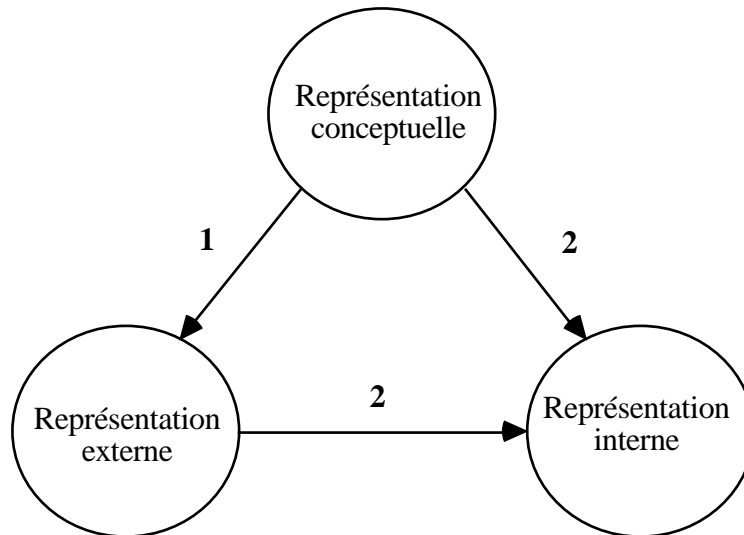


A la fin de l'étude de l'existant, nous disposons donc au minimum d'une description des procédures prévues et minimales des différents postes de travail qui vont faire l'objet d'une nouvelle conception.

5.2 Conception d'une application interactive

Nous exposons dans ce paragraphe comment concevoir les nouveaux logiciels interactifs à partir des spécifications de l'existant. Pour chaque poste de travail, cette conception se subdivise en trois parties; nous concevons d'abord la Représentation Conceptuelle de

l'application puis la Représentation Externe et en dernier la Représentation Interne qui dépend à la fois des deux autres représentations.



Nous n'exposons pas ici la méthode de conception de la Représentation Interne car celle-ci est trop dépendante des logiciels de base disponibles et des méthodes de programmation; elle nécessite à elle seule un développement égal à l'ensemble de ce livre. Nous présentons donc ci-dessous la méthode de conception de la Représentation Conceptuelle et de la Représentation Externe.

5.2.1 Méthode de conception de la Représentation Conceptuelle

Nous allons définir, dans le premier paragraphe, les trois types de procédures (prévue, minimale, effective) du nouveau système et dans le deuxième paragraphe, le détail d'une procédure.

5.2.1.1 Les différentes procédures de la Représentation Conceptuelle

L'ordre d'élaboration de ces procédures peut varier selon les cas :

- si les nouvelles procédures automatisées sont proches des procédures existantes, on définira en premier lieu les procédures prévues (puis minimale puis effective).
- si les nouvelles procédures automatisées sont éloignées des procédures existantes, on définira en premier lieu la procédure minimale (puis prévue puis effective).

Ces trois types de procédures n'ont pas du tout le même statut méthodologique; *il est indispensable de définir la procédure minimale; il est confortable de définir une procédure prévue; il peut être utile et efficace de définir une ou des procédures effectives* .

Procédure minimale

Il y a deux manières différentes de la définir selon que le nouveau système est plus ou moins éloigné de l'ancien système.

Premier cas

Elle est déduite de la procédure minimale existante si celle-ci est peu ou pas modifiée par l'automatisation. Dans ce cas, nous repartons de cette procédure minimale existante et pour chaque opération, nous nous demandons si cette opération reste manuelle ou devient interactive ou automatique.

Pour la procédure constituée de ces opérations interactives et automatisées, nous nous posons la question fondamentale suivante:

"Les précédences permanentes et les opérations obligatoires du système existant doivent-elles être contrôlées par l'ordinateur?"; ou en d'autres termes, "transfère-t-on une partie du contrôle de l'organisation à l'ordinateur?".

La plupart du temps, la réponse est positive, car on pense que ce transfert ne peut qu'accroître l'efficacité du contrôle.

Mais ce transfert n'est pas toujours aussi direct car on profite souvent de l'automatisation d'une procédure pour procéder à des réorganisations qui ont justement un impact sur la latitude décisionnelle des différents postes de travail: on peut être amené à augmenter ou à diminuer cette latitude décisionnelle, ce qui influe sur la nature des précédences, des opérations et des déclenchements.

Il faut noter que ces modifications sont souvent faites implicitement sans que le lien soit fait explicitement avec l'organisation.

L'objectif essentiel de cette étape est de rendre explicites les choix organisationnels et leurs conséquences sur le niveau décisionnel de chaque poste de travail.

Pour déterminer les déclenchements systématiques, il faut se demander si "l'enchaînement est immédiat ou non entre deux opérations reliées par une précedence permanente".

Si l'enchaînement est immédiat, on a un déclenchement systématique, c'est-à-dire déclenché par l'ordinateur. Dans tous les autres cas, on a un déclenchement optionnel, c'est-à-dire contrôlé par l'utilisateur.

Au niveau de la procédure minimale, il vaut mieux prévoir le maximum de déclenchements optionnels pour avoir la plus grande latitude dans la définition des procédures prévues et effectives.

Deuxième cas:

La procédure minimale est définie "ex nihilo" si l'automatisation modifie trop la procédure minimale existante.

Pour cela, on part de l'ensemble des opérations appartenant au but considéré et pour chaque opération nous définissons, comme précédemment, son caractère obligatoire ou facultatif et la nature de son déclenchement. Nous introduisons ensuite les précédences permanentes correspondant aux règles de gestion non transgressables .

La procédure prévue

Elle correspond au cas normal ou usuel. Son rôle est d'une part de servir de guidage à un utilisateur novice qui se poserait une question du type "comment faire pour atteindre le but x", d'autre part de simplifier le travail de l'utilisateur en lui fournissant une procédure réduite au cas normal avec le maximum d'aide et d'enchaînement automatique.

Si la conception du système automatisé n'entraîne pas ou peu de modifications, la procédure prévue peut être déduite directement de la procédure prévue existante telle qu'elle a été recueillie lors de l'étude de l'existant .

Si la conception du système automatisé entraîne une modification de la logique de fonctionnement du système d'information et/ou une réorganisation, il faudra la créer à partir de la procédure minimale. Pour cela, on rajoutera à la procédure minimale des précédences indicatives et des déclenchements automatiques qui rendront l'exécution de la procédure prévue plus commode et plus rapide. Bien entendu, il ne s'agira seulement que d'hypothèses sur le fonctionnement usuel de la procédure qui demanderont à être validées auprès des utilisateurs sur un prototype ou sur le logiciel final.

La procédure effective

Les procédures effectives ne sont pas nécessairement définies à ce stade de la méthode. En effet, une procédure effective ne peut être définie ici que si l'on a observé lors de l'étude de l'existant des variantes de procédures liées à des types différents d'utilisateurs (novices, expérimentés, occasionnels...).

Dans le cas où on a pu procéder à ces observations, on définira une procédure effective en rajoutant des précédences indicatives à la procédure minimale qui correspondent à l'usage de l'utilisateur, en augmentant les déclenchements automatiques pour minimiser les manipulations de l'utilisateur et en supprimant des opérations facultatives non nécessaires dans ce cas particulier.

Si l'on n'a pas pu procéder à ces observations, on ne pourra pas définir de procédure effective à ce stade de la méthode. Il faudra alors attendre d'avoir réalisé complètement l'application interactive (sous sa forme minimale et prévue) afin de l'expérimenter auprès des utilisateurs et de rajouter ensuite à la demande différentes procédures effectives (ce qui suppose une structure de logiciel permettant cette adaptation telle que nous l'avons définie dans le chapitre 4).

5.2.1.2 Les éléments d'une procédure

Paramètres constants

Les paramètres constants désignent l'ensemble des aides offertes à l'utilisateur indépendamment de toute application.

Dans le chapitre 2, nous avons défini trois types de paramètres constants; les premiers concernent l'aide au travail (interrompre, quitter, annuler, transférer,...), les seconds l'aide à l'apprentissage (guidage fonctionnel et guidage d'utilisation), les troisièmes les possibilités d'évolution (ajout de procédures effectives,...).

A ce stade du déroulement de la méthode, il faut décider quels sont les paramètres constants qui seront effectivement mis systématiquement à la disposition des utilisateurs.

Ce choix peut dépendre de la nature du travail, de la latitude décisionnelle des utilisateurs, mais aussi de contraintes matérielles et logicielles.

Ces choix ont des conséquences sur la Représentation Conceptuelle des paramètres variables de la procédure.

En effet, si on ne donne pas à l'utilisateur, par exemple, les possibilités systématiques de quitter ou d'interrompre, il faudra prévoir dans la description de la procédure les endroits spécifiques où l'utilisateur pourra activer ces commandes.

Paramètres variables

Les paramètres variables correspondent à la description des procédures spécifiques d'une application.

La méthode Diane pouvant se greffer sur la méthode Merise, nous utilisons le formalisme graphique de Merise pour la description des

événements, des opérations, des règles d'émission et des synchronisations.

Nous rajoutons à ce formalisme différents concepts qui ont pour but de définir la répartition du pilotage entre l'homme et l'ordinateur dans le cadre des applications interactives et qui permettent de répondre aux questions suivantes:

- Qui déclenche l'opération?

Si c'est l'utilisateur, on a un déclenchement optionnel; si c'est l'ordinateur, c'est un déclenchement automatique.

- Qui fait l'opération?

Si c'est l'utilisateur, c'est une opération manuelle; si c'est l'ordinateur, c'est une opération automatique; si c'est les deux, c'est une opération interactive.

- Qui contrôle l'exécution de l'opération?

Si c'est l'utilisateur, c'est une opération facultative; si c'est l'ordinateur, c'est une opération obligatoire.

- Qui contrôle l'enchaînement entre opérations?

Si c'est l'utilisateur, c'est une précedence indicative; si c'est l'ordinateur, c'est une précedence permanente.

L'ensemble de ces paramètres variables ainsi que le formalisme associé est présenté dans la deuxième partie du chapitre 2.

L'étude de cas présentée à la fin de ce chapitre illustre de façon détaillée la description des procédures de la Représentation Conceptuelle.

5.2.1.3 Démarche globale pour la Représentation Conceptuelle

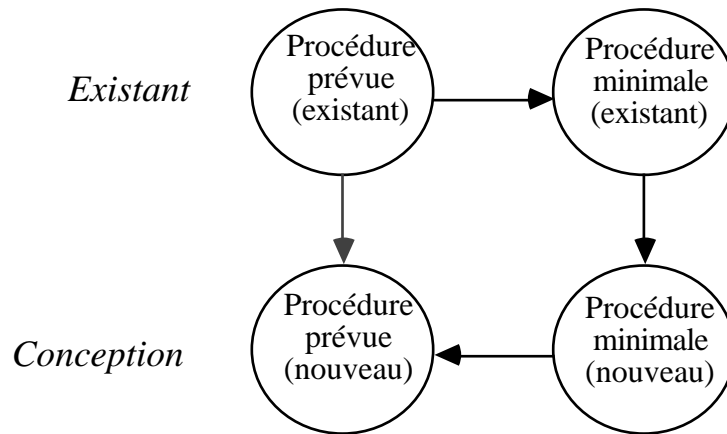
Pour mieux visualiser la méthode de conception, nous représentons sur les deux schémas suivants deux façons de la mettre en oeuvre selon le contexte de l'étude et de l'entreprise. Dans les deux cas, nous supposons qu'il y a un existant qui sert de base à la nouvelle conception.

Mais cette phase d'étude de l'existant peut ne pas avoir lieu, soit parce que on a affaire à un nouveau service ou une nouvelle entreprise, soit parce que l'on a décidé de modifier en profondeur le système existant et qu'il ne doit plus servir de référence. Les schémas que nous présentons se réduisent alors à la phase de conception suivie éventuellement de la phase d'expérimentation.

Démarche minimale

Le premier schéma montre la démarche minimale où l'on part de la procédure prévue existante recueillie par entretien ou questionnaire pour extraire la procédure minimale existante par entretien à partir de la procédure prévue.

La conception se fait alors par le chemin inverse avec la détermination de la nouvelle procédure minimale à partir de la procédure minimale existante puis la conception de la nouvelle procédure prévue en fonction de la procédure prévue existante et de la nouvelle procédure minimale.



Cette démarche est minimale en ce sens qu'on ne peut pas réduire la longueur de l'étude à un niveau inférieur en temps et en informations recueillies.

Elle peut éventuellement suffire si les procédures sont simples ou si le niveau décisionnel du poste de travail est bas.

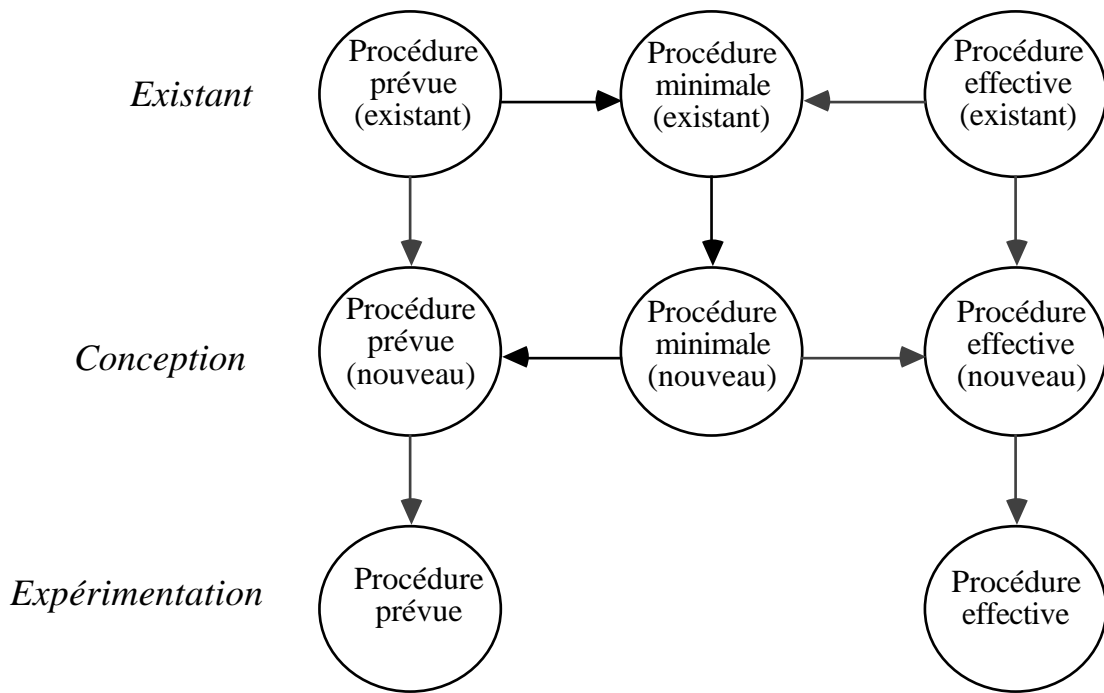
Dans les autres cas, il vaut mieux faire une étude plus complète comme nous le présentons ci-dessous.

Démarche maximale

L'étude de l'existant peut être complétée par le recueil d'une ou de plusieurs procédures effectives qui servent à déterminer de nouvelles procédures effectives compte-tenu de la nouvelle procédure minimale.

Ultérieurement, lors de la phase d'expérimentation (après la détermination de la Représentation Externe), on peut tester l'adéquation de la nouvelle procédure prévue en tant que guide pour des utilisateurs novices et l'adéquation des procédures effectives pour les utilisateurs expérimentés.

Cette démarche est beaucoup plus coûteuse en temps d'analyste et d'utilisateur car les deux parties que nous rajoutons (procédures effectives et expérimentations) font appel à des recueils de trace de travail (observations, autorelevés, capteurs).



Les flèches en trait plein indiquent la démarche obligatoire minimale; les flèches en pointillé indiquent une démarche possible.

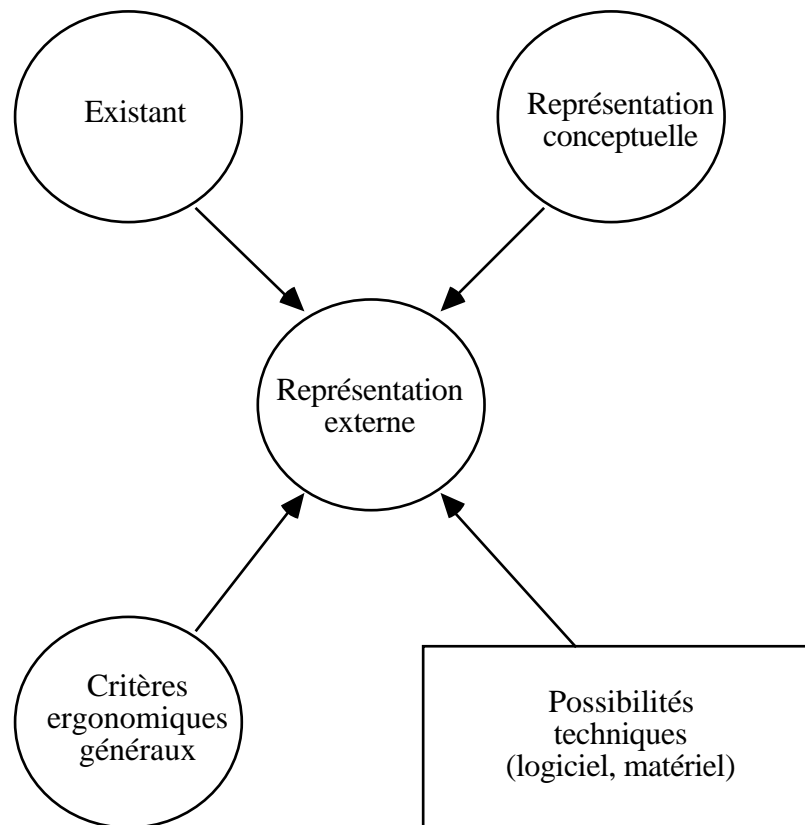
5.2.2 Méthode de conception de la Représentation Externe

La conception de la Représentation Externe est dépendante de plusieurs éléments que nous avons schématisés sur la figure ci-dessous.

En premier lieu, elle comprend la traduction des spécifications déterminées dans la Représentation Conceptuelle.

Elle intègre aussi certains éléments d'ergonomie provenant de l'étude de l'existant comme le vocabulaire spécialisé, ou ayant un caractère général comme l'homogénéité.

Et elle dépend également des possibilités techniques matérielles et logicielles des ordinateurs sur lesquels se fait l'implémentation.



5.2.2.1 Traduction de la Représentation Conceptuelle

Dans la première étape de la définition de la Représentation Externe il s'agit de donner une représentation externe aux paramètres définis dans la Représentation Conceptuelle et que nous avons détaillés à la fin du chapitre 2.

On distingue, comme précédemment, les paramètres constants des paramètres variables qui sont propres à l'application.

Les paramètres constants de la Représentation Conceptuelle concernent l'aide à l'utilisation, la mémorisation et le guidage; ces paramètres doivent être traduits sous forme de commandes activables par l'utilisateur.

Pour chacune de ces commandes, il faut prévoir:

- un nom sans ambiguïté pour les utilisateurs et compatible avec les logiciels existants
- une forme de présentation (caractères gras, italiques,...) éventuellement variable pour indiquer que la commande est ou n'est pas activable
- un mode de désignation (touche-fonction, souris,...) et une syntaxe rapide et facile à mémoriser
- une détection d'erreurs et le contenu des messages correspondants

- une subdivision, s'il y a lieu, correspondant à une fréquence d'utilisation.

Les paramètres propres à l'application se divisent en paramètres provenant de la procédure qui se traduisent sous la forme de commandes activables par l'utilisateur et de paramètres provenant des opérations qui se traduisent en données.

Pour les commandes spécifiques à l'application, il faut prévoir:

- une hiérarchie de l'ensemble de ces commandes qui reflète la logique d'utilisation
- une présentation des commandes (menu déroulant, glissant, fixe) et une forme de caractères avec éventuellement une deuxième forme pour distinguer les commandes activables des autres (menu dynamique)
- la dénomination de ces commandes en utilisant au maximum le vocabulaire et les abréviations des spécialistes s'ils existent.
- le mode de désignation et la syntaxe
- le contenu des messages d'erreurs qui peut être très variable selon que l'on décide ou non de distinguer les erreurs d'intention des erreurs d'exécution.

La traduction des opérations amène la définition des éléments suivants:

- la présentation de la zone de dialogue avec la distinction des zones d'entrée (saisie de l'utilisateur) et des zones de sortie (réponse de l'ordinateur)
- le vocabulaire des données issu du vocabulaire des spécialistes
- la syntaxe de fin de saisie de données et de validation de zones ou d'écrans
- le contenu des messages d'erreur en fonction des erreurs que l'on a choisi de détecter et des possibilités de correction.

La traduction de l'ensemble des paramètres que nous venons d'énumérer doit se faire en tenant compte d'une part des critères ergonomiques, d'autre part des possibilités techniques, comme nous allons le détailler dans les deux paragraphes suivants.

5.2.2.2 Prise en compte des critères ergonomiques

Nous distinguons les critères ergonomiques à caractère général des éléments ergonomiques provenant de l'étude de l'existant, c'est-à-dire propres à l'application.

Le premier critère à intégrer à la Représentation Externe est le critère d'homogénéité qui a des conséquences sur trois éléments:

- la présentation générale des écrans, c'est-à-dire le choix des emplacements des zones de menu, de saisie de données, de messages d'erreur, de messages de services, cette présentation devant être homogène sur l'ensemble de l'application et si possible avec les autres logiciels existants;
- les dispositifs d'entrée en distinguant les dispositifs d'entrée des données de ceux d'entrée des commandes et, éventuellement, en distinguant les commandes standard et les commandes particulières; l'entrée des commandes standard doit être homogène sur l'ensemble du logiciel, rapide et facile à mémoriser; le dispositif d'entrée des commandes particulières doit être aussi homogène et doit se distinguer sans ambiguïté du dispositif d'entrée des données;
- la syntaxe du langage de commande, c'est-à-dire la validation des commandes, des données et des écrans.

Les éléments provenant de l'existant se répartissent en deux ensembles; les premiers ont déjà été utilisés au niveau de la conception de la Représentation Conceptuelle: il s'agit de la répartition du pilotage entre l'homme et la machine et de la logique d'utilisation; les seconds sont intégrables uniquement au niveau de la Représentation Externe et il s'agit:

- du vocabulaire des spécialistes qui sert à déterminer le vocabulaire des données et des commandes quand cela est possible, c'est-à-dire quand il y a correspondance entre les opérations existantes et les opérations créées par l'automatisation;
- de la présentation des formulaires ou des écrans existants dans la mesure où il n'y a pas de modifications du système d'information existant;
- des erreurs à détecter et des possibilités de correction qu'il y a à faire compte-tenu des impératifs de fiabilité de l'application et des types d'erreurs usuels commis.

5.2.2.3 Prise en compte des possibilités techniques

La détermination de la Représentation Externe est elle aussi fortement dépendante des possibilités matérielles et logicielles des ordinateurs sur lesquels se fait la réalisation.

Sur le plan matériel, il y a deux contraintes:

- la définition de l'écran qui donne la limite du nombre de caractères aisément lisibles par l'utilisateur et les possibilités graphiques;

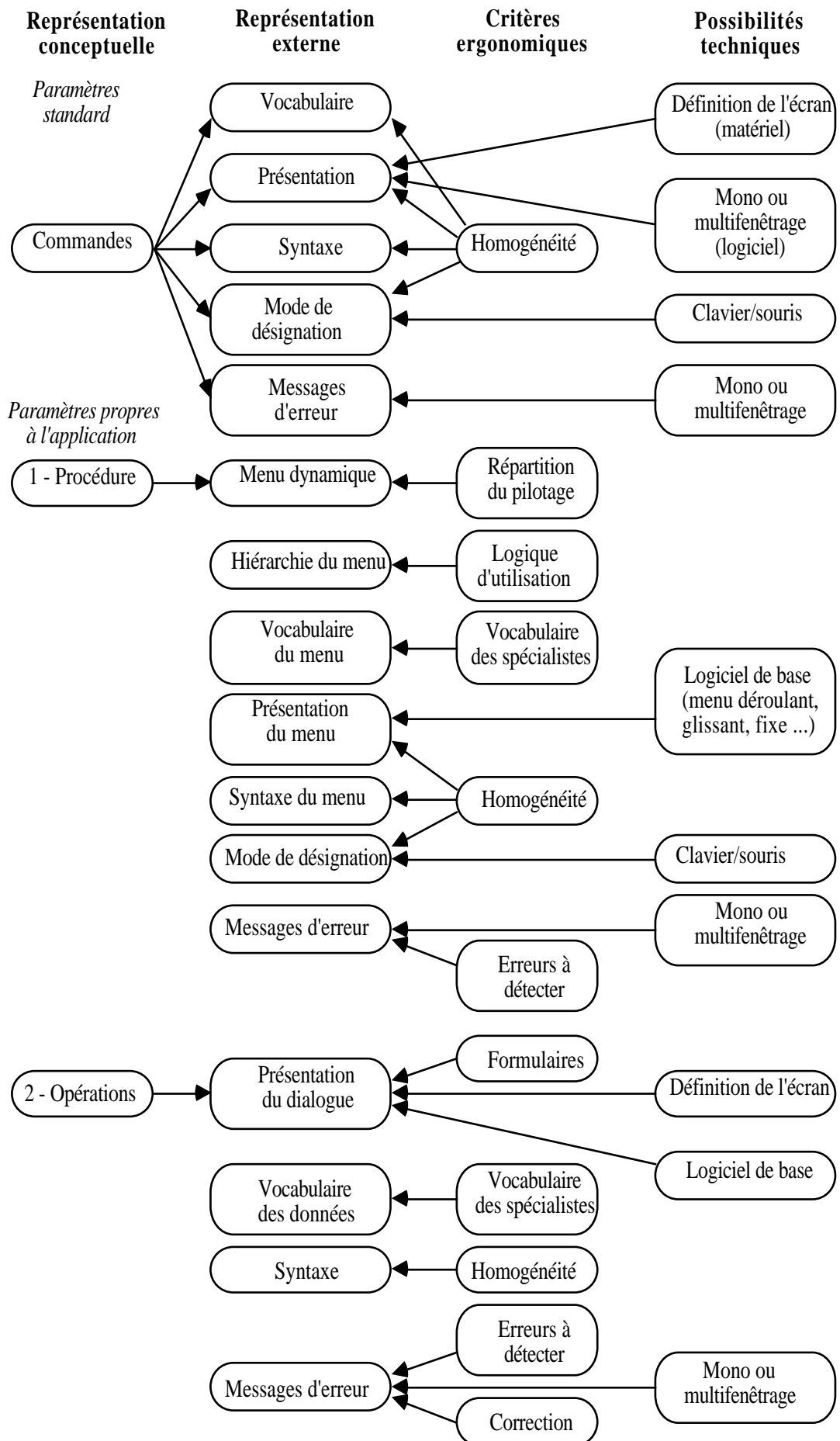
- les dispositifs physiques d'entrée (clavier, souris,...) qui offrent plus ou moins de choix pour distinguer par exemple les commandes des données.

Sur le plan logiciel, la contrainte essentielle tient au type de logiciel de gestion d'écran qui impose des contraintes très fortes sur la présentation générale et particulière de l'interaction ainsi que sur les possibilités de commandes standard comme l'interruption. Cet élément a été développé dans les chapitres 3 et 4.

La conception de la Représentation Externe ne peut se faire de manière linéaire en intégrant d'abord la Représentation Conceptuelle, puis les critères ergonomiques, puis les possibilités techniques; ces trois éléments doivent être intégrés simultanément pour la conception de chacun des paramètres de la Représentation Externe.

Nous avons visualisé ces interactions dans le schéma suivant.

Conception de la RE (synthèse)



5.3 Prototypage

L'objectif de cette étape est de montrer comment faire des tests auprès des utilisateurs sur le logiciel conçu et réalisé dans les trois étapes précédentes.

Si la réalisation a abouti directement au logiciel définitif, les modifications consécutives aux tests que nous allons pratiquer dans cette phase risque d'être coûteux en temps de programmation. Aussi est-il préférable, quand cela est possible, de procéder aux tests sur des versions du logiciel non définitives, c'est-à-dire sur des maquettes ou des prototypes.

Nous appelons *maquette* une simulation partielle du logiciel; certains outils de génie logiciel engendrent automatiquement des maquettes qui permettent par exemple de simuler l'enchaînement des écrans. Les maquettes sont surtout utiles à l'analyste car elles lui permettent de visualiser les conséquences des spécifications qu'il vient de faire, mais elles sont d'un intérêt limité pour des tests auprès des utilisateurs car trop éloignées du logiciel définitif.

Nous appelons *prototype* un logiciel qui possède toutes les fonctionnalités du logiciel définitif, mais sans souci de performances (optimisation des temps de réponse et de la place mémoire). Les prototypes peuvent être générés automatiquement par un outil de génie logiciel ou écrits directement avec un langage d'intelligence artificielle (PROLOG, SMALLTALK,...). Les prototypes correspondent à une version du logiciel définitif réaliste qui permettent de réaliser efficacement des tests auprès des utilisateurs, puis de le modifier à un moindre coût.

Cette étape de tests auprès des utilisateurs n'est pas encore couramment pratiquée en informatique, mais dans le cas de logiciels interactifs elle est indispensable car plusieurs éléments de la Représentation Externe ne peuvent être déterminés en l'absence de l'utilisateur.

5.3.1 Éléments à tester

Les éléments à tester sont d'une part des éléments simples pour lesquels il n'y a pas de critères ergonomiques d'autre part l'interaction d'éléments simples dont l'effet est a priori non prévisible.

Éléments simples

Il s'agit ici de tester tous les éléments de la Représentation Externe qui n'ont pas pu être défini à partir de critères ergonomiques:

- le *vocabulaire* qui n'est pas inspiré du vocabulaire des spécialistes; c'est généralement le cas du vocabulaire de commandes correspondant à des opérations propres à l'ordinateur ou à de nouvelles opérations qui n'existaient pas;
- la *présentation des zones de saisie* de données nouvelles qui ne correspondent pas à des formulaires ou des écrans existants;
- la *logique d'utilisation* pour des tâches nouvelles ou profondément modifiées par l'automatisation;
- les *procédures effectives* mises en oeuvre par les utilisateurs sur les nouveaux logiciels.

Interaction d'éléments simples

Nous avons présenté au chapitre 3 des critères ergonomiques pour chacun des paramètres de la Représentation Externe, mais quand on veut appliquer l'ensemble de ces critères sur une application particulière avec des possibilités techniques particulières, on est amené à faire des choix, des compromis qui n'optimisent pas forcément l'ensemble des paramètres. C'est pour pouvoir tester les effets de ces choix qu'une expérimentation sur prototype est également nécessaire.

Nous citons ci-dessous les interactions les plus courantes:

- pour la présentation du dialogue, il peut s'avérer impossible pour des raisons techniques (définition de l'écran) de suivre la mise en page des formulaires existants; on doit alors proposer une présentation des données cohérente avec un ordre "standard" de saisie et l'expérimenter;

- pour le mode de désignation, il peut y avoir incompatibilité entre le critère d'homogénéité et les possibilités techniques des dispositifs d'entrée.

Par exemple, on peut être limité par le nombre de touches-fonctions disponibles par rapport au nombre de commandes standard ou propres à l'application; il faut alors choisir de façon arbitraire les commandes qui seront activées par touche-fonction (de préférence les commandes standard) et choisir également un autre mode de désignation pour les autres commandes;

- il peut y avoir incompatibilité entre la hiérarchie du menu et sa présentation; en effet, la hiérarchie du menu doit refléter la logique d'utilisation et la présentation du menu est dépendante des possibilités techniques et doit être homogène sur l'ensemble des logiciels.

Pour une application particulière, on peut avoir un niveau du menu pour lequel les libellés des commandes correspondantes soient en nombre supérieur à la capacité de la zone de présentation du menu.

Il faut alors choisir entre la modification de la présentation du menu et la modification de la hiérarchie de ce menu;

- le mode de désignation peut impliquer des contraintes pour le choix du vocabulaire des commandes.

En effet, si la désignation n'est pas directe (saisie de la commande au clavier), il faut prévoir une abréviation de cette commande. Cette abréviation doit être l'application d'une règle mnémotechnique sur les noms de commande; les noms de commandes sont par ailleurs choisis soit dans le vocabulaire des spécialistes, soit pour leur non-ambiguïté. Il n'est pas évident qu'en appliquant la règle d'abréviation sur ces noms, on obtienne des abréviations toujours différentes.

Dans ce cas, il faut choisir entre modifier le vocabulaire des commandes ou modifier la règle d'abréviation.

5.3.2 Comment tester

On peut procéder à des tests "en laboratoire" ou à des tests "sur le terrain".

Les tests en laboratoire sont surtout utiles sur une première version du prototype pour en éliminer les dysfonctionnements les plus criants, mais les tests sur le terrain sont plus révélateurs. Cependant ces derniers ne sont pas toujours possibles car ils peuvent s'avérer perturbants pour le milieu de travail.

Si cela est possible, il est intéressant de procéder à des tests à deux moments: tout d'abord en début d'utilisation, puis un ou deux mois après le début de l'utilisation.

Les tests de début permettent surtout de tester les facilités d'apprentissage alors que les tests ultérieurs permettent de tester les facilités d'utilisation pour des utilisateurs expérimentés.

Les tests sont obligatoirement longs pour l'analyste car il n'est pas possible de procéder à de simples entretiens comme dans le cas de l'étude de l'existant.

En effet, au cours d'un entretien, l'utilisateur fait une reconstruction mentale du travail qu'il a effectué et cette reconstruction a tendance à gommer ou à surévaluer les erreurs qu'il commet et les difficultés qu'il rencontre.

Les seuls moyens de tests fiables dans ce contexte sont les observations et les capteurs (les autorelevés étant trop coûteux en temps utilisateur).

Les capteurs sont particulièrement bien adaptés aux tests sur prototype car ils peuvent être rajoutés au logiciel du prototype sans fausser le cours normal de l'exécution du travail de l'utilisateur.

Le capteur le plus simple consiste en la mémorisation systématique de l'ensemble des transactions effectuées entre l'utilisateur et le logiciel. Mais ce type de capteur fournit une masse d'informations difficile à synthétiser; aussi est-il souvent plus judicieux de faire des capteurs plus sélectifs portant sur les éléments qu'il nous semble prioritaire de tester, ce qui facilite les synthèses ultérieures.

La phase de tests à partir de capteurs peut être précédée ou suivie d'observations par un tiers ou par l'analyste.

Les observations faites avant permettent de dégager les éléments sur lesquels il sera nécessaire de faire des tests au moyen des capteurs.

Dans tous les cas, les observations permettent de dégager des faits qui ne peuvent être enregistrés par capteurs; c'est le cas par exemple de toutes les consultations d'informations hors logiciels effectuées par les utilisateurs.

Ces observations peuvent être complétées par des entretiens portant sur certains faits observés et difficiles à interpréter; il peut être utile de savoir pourquoi telle information a manqué à un moment particulier.

5.4 Etude de cas "Bibliothèque"

"Bibliothèque" est une étude complète qui part de la description du système d'information existant selon MERISE pour aboutir à la représentation externe correspondant à une opération donnée .

Le but essentiel étant d'illustrer les méthodes présentées dans l'ouvrage, l'exemple a été choisi volontairement simple et n'exige du lecteur aucune compétence technique particulière dans le domaine de la documentation, hormis la connaissance du concept de thesaurus .

Thesaurus : liste alphabétique de mots-clés reliés par des liens sémantiques (synonymie, homonymie, hiérarchie...) ; permet de rechercher un ouvrage non seulement par le titre ou l'auteur, mais aussi à partir de mots-clés .

Nous présentons d'abord la description du système existant avec le formalisme MERISE, puis la conception du nouveau système.

5.4.1 Le système d'information existant

5.4.1.1 Les fonctions

Les schémas qui suivent décrivent en détail le système d'information existant, selon une classification en fonctions.

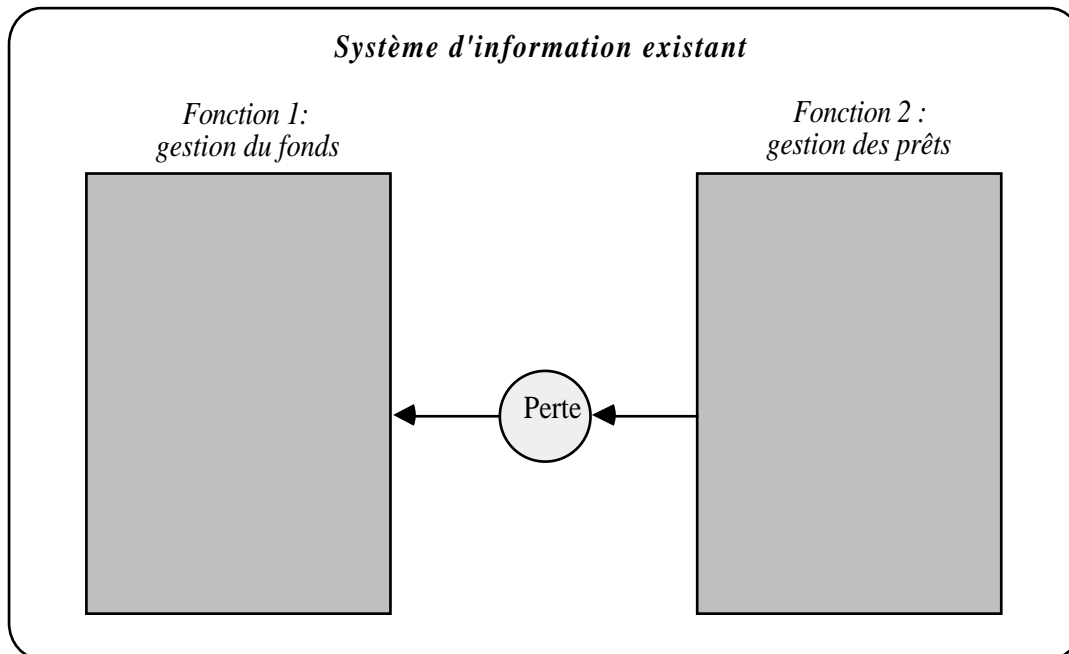


Fig 1.1

On distingue, dans l'existant, deux grandes fonctions (figures 1.3 et 1.4):

- gestion du fonds documentaire (commander et enregistrer les livres ...)
- gestion des prêts de livres aux abonnés (rechercher les ouvrages, enregistrer les prêts ...).

Cette distinction revient à partager le système en deux sous-systèmes. Le choix du découpage se justifie, au sens de la théorie des systèmes, par le fait que les inter-relations entre sous-systèmes sont ainsi réduites au minimum : les deux fonctions présentent un seul point d'intersection, matérialisé par la circulation d'une fiche en cas de perte d'ouvrage.

5.4.1.2 Utilisateurs

Trois types d'utilisateurs sont confrontés au système d'information :

- l'administratif intervient dans l'achat et le paiement des livres ;
- le documentaliste enregistre et prête les livres ;
- l'abonné de la bibliothèque consulte et emprunte les livres .

Le découpage en deux sous-systèmes est tel que l'administratif et l'abonné n'ont affaire qu'à un des deux sous-systèmes alors que le documentaliste intervient dans les deux. Notons que la gestion des prêts fait parfois intervenir simultanément l'abonné et le documentaliste .

L'interaction entre le découpage en fonctions et en postes de travail est présentée sur le schéma suivant:

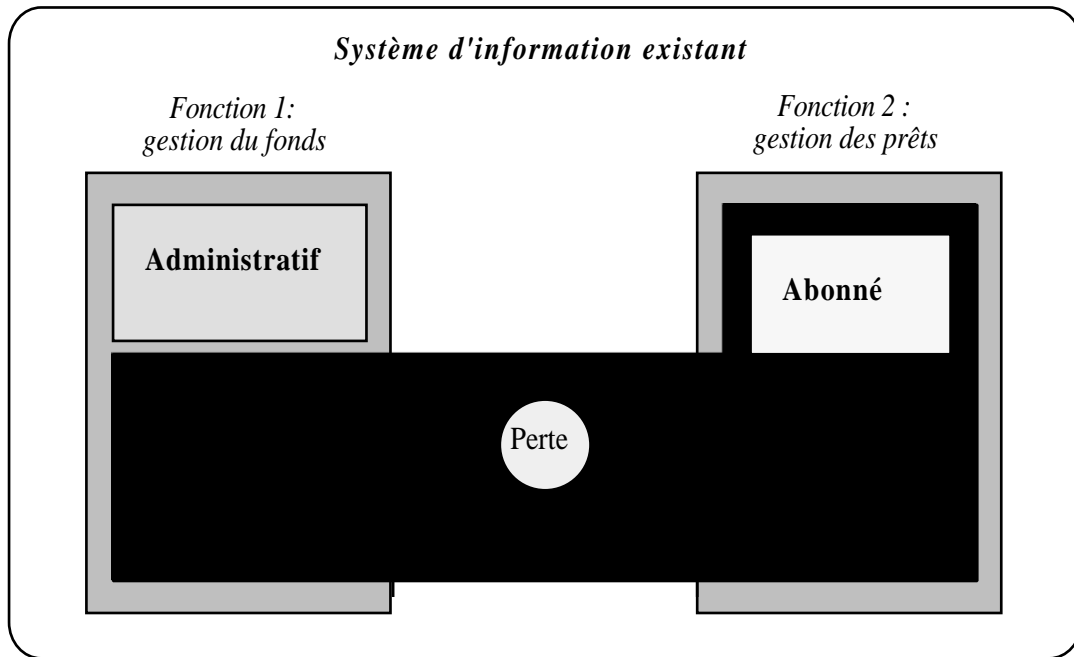


Fig1.2

Existant

Fonction 1 : gestion des fonds

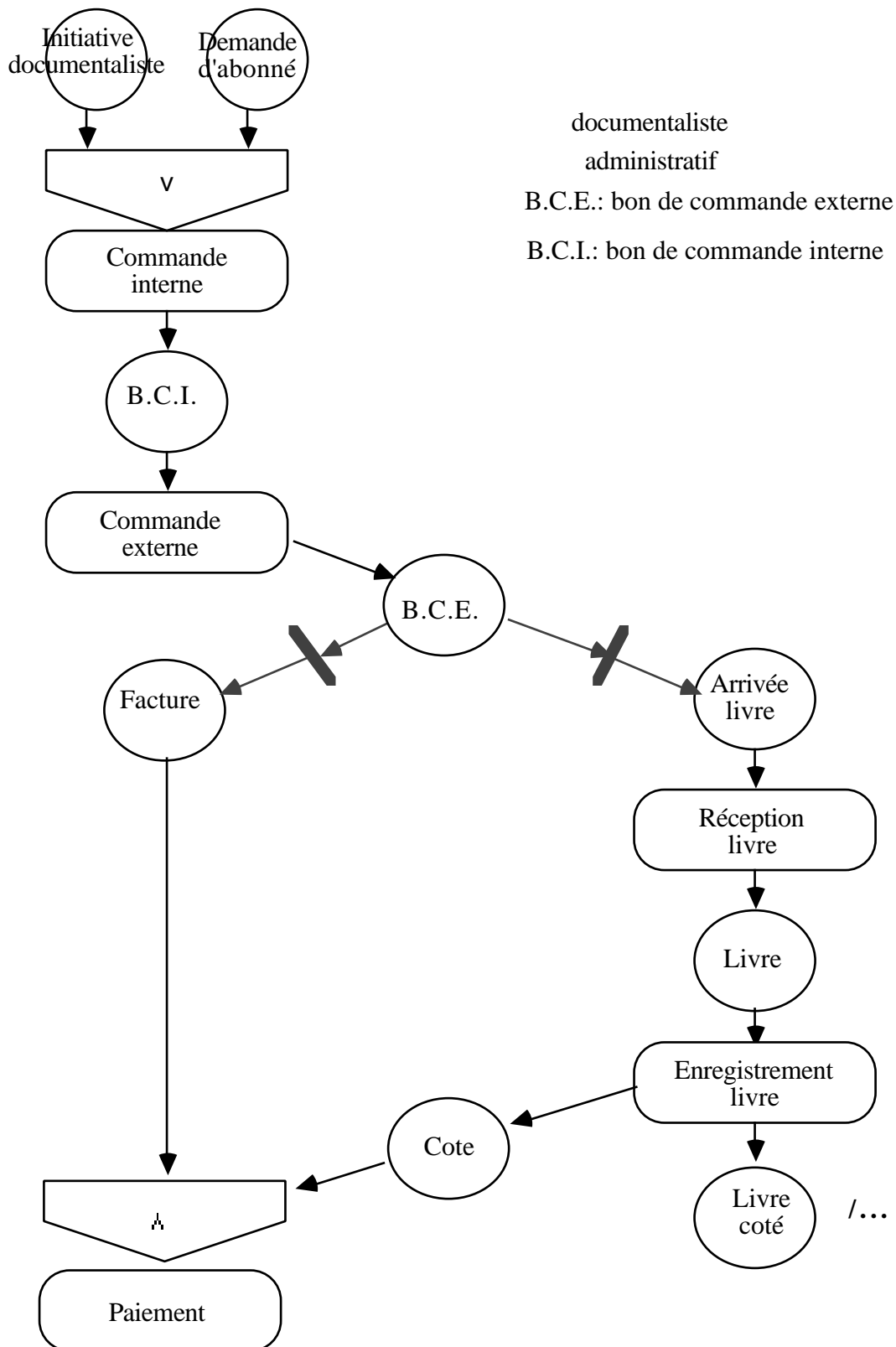


Fig 1.3

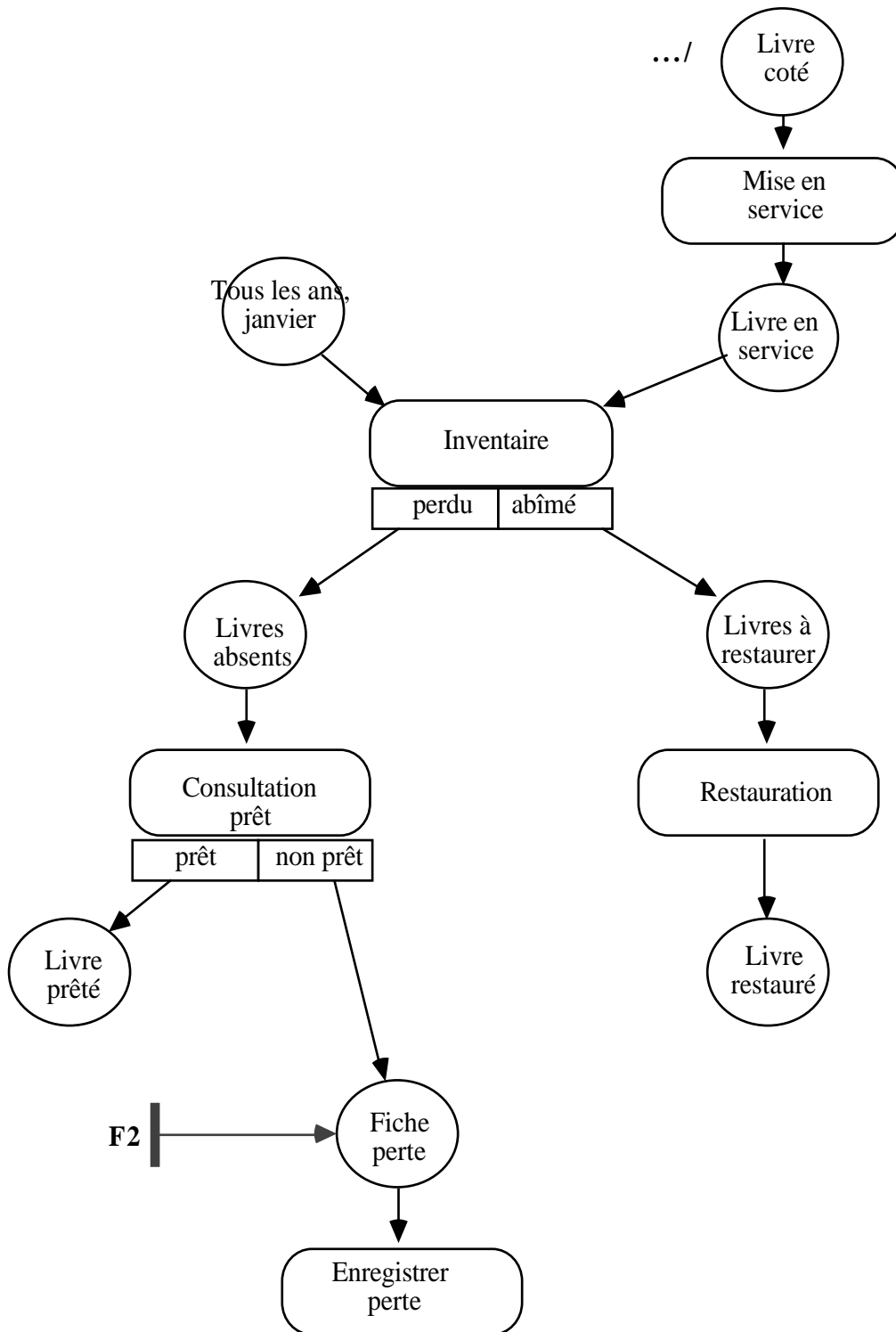
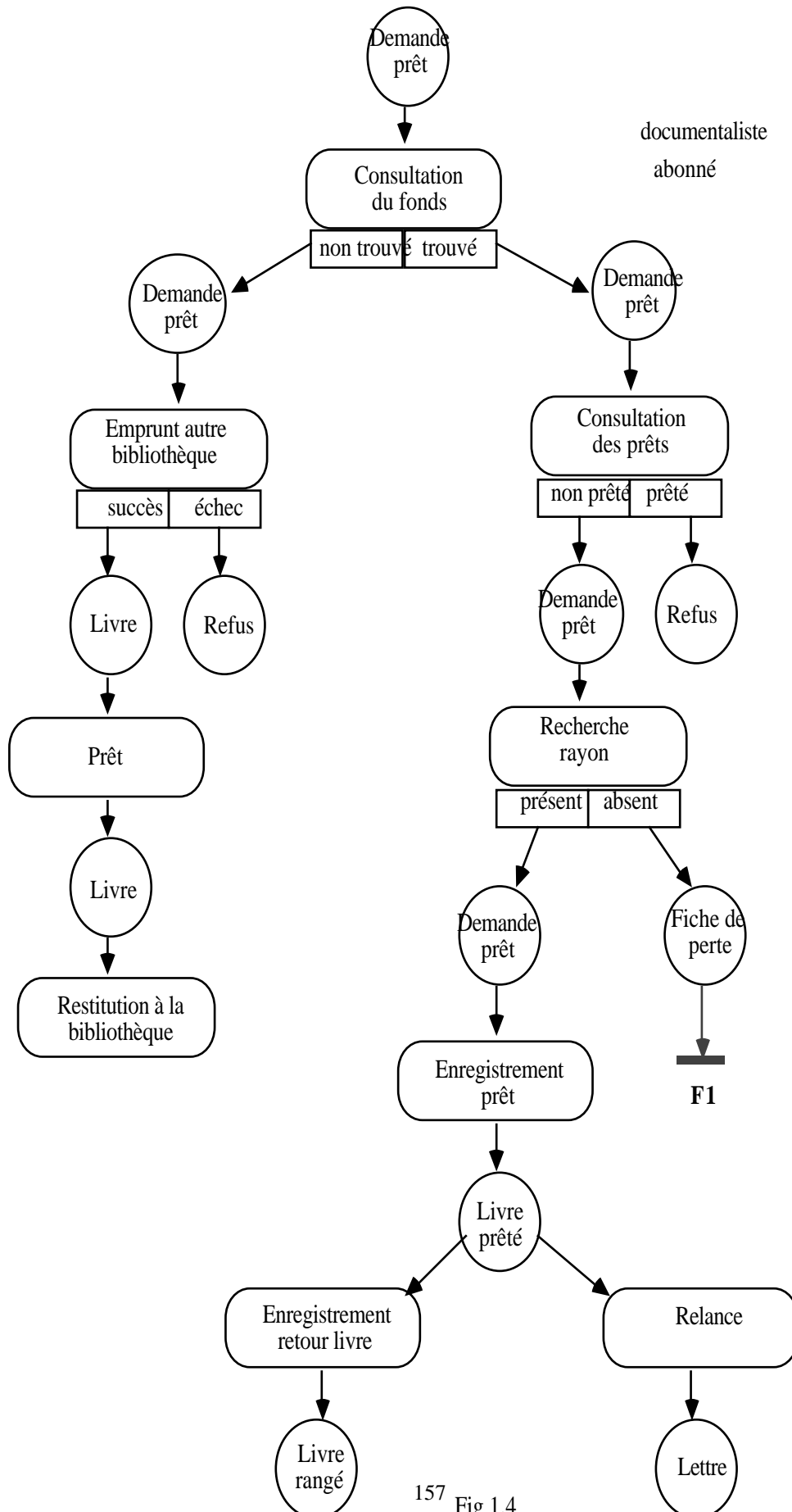


Fig 1.3 (suite)

Existant

Fonction 2 : gestion des prêts



Nous montrons à titre d'exemple une seule procédure minimale du système existant car les procédures minimales du système existant et du nouveau système sont très proches et ne se différencient que par l'indication des opérations interactives et automatiques dans le nouveau système.

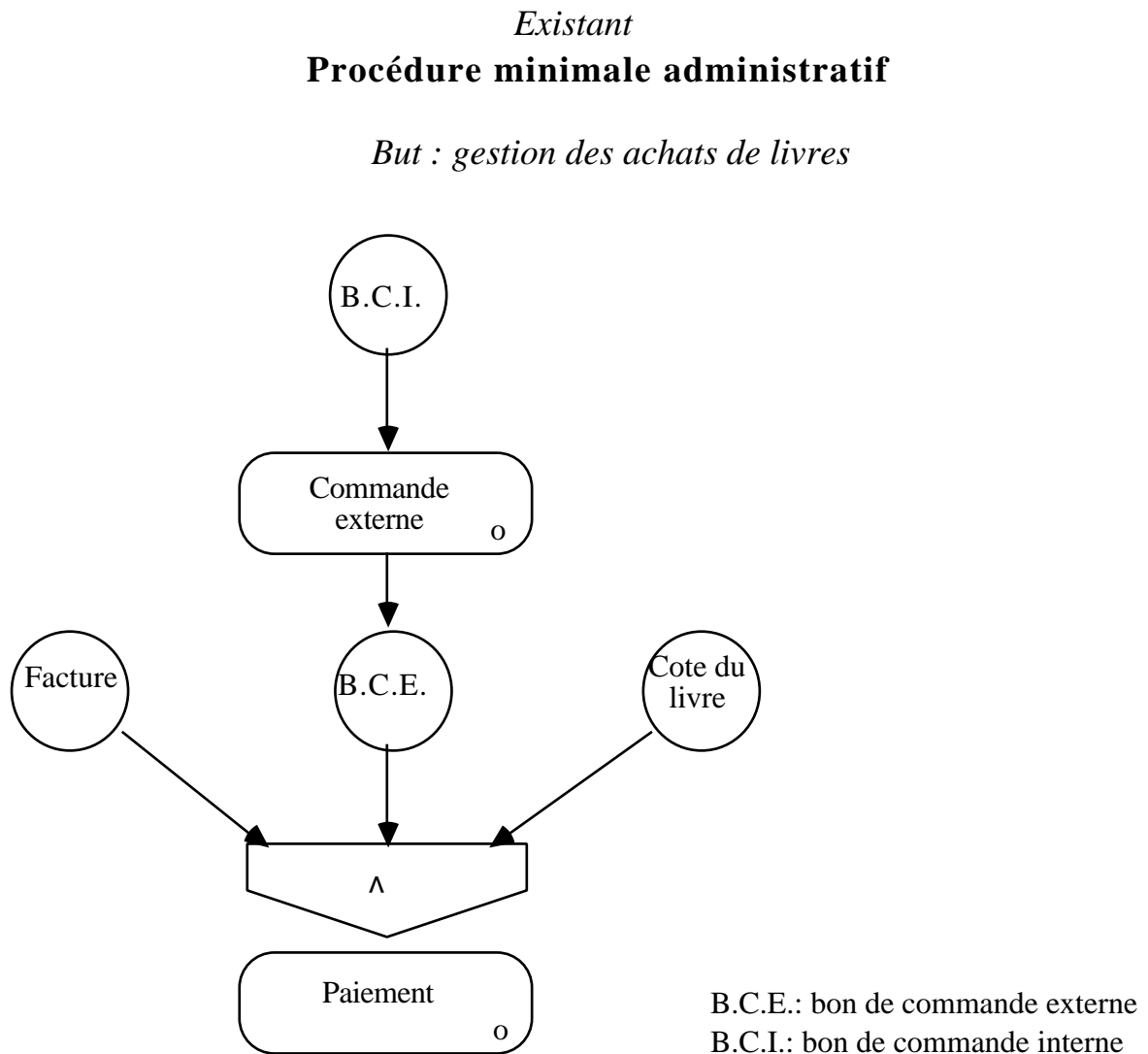


Fig 1.5

5.4.2 Conception du nouveau système

5.4.2.1 Conception de la Représentation Conceptuelle

Nous présentons les éléments suivants:

- le schéma de la base de données selon le formalisme entité-relation de MERISE (figure 1.6)

- la procédure prévue de l'abonné, décrite par la figure 1.9
- les procédures minimales pour tous les postes de travail: la procédure minimale de l'administratif est décrite par la figure 1.7 et celle de l'abonné par la figure 1.8. Pour le documentaliste, il y a trois procédures minimales correspondant à trois buts différents de ce poste de travail ; elles sont décrites par les figures 1.10 à 1.14;
- une procédure effective du poste du documentaliste pour deux buts différents (gestion du fonds propre et gestion des prêts), décrite par les figures 1.15 et 1.16
- le détail d'une opération (enregistrement prêt), décrit par la figure 1.18.

Schéma de la base de données

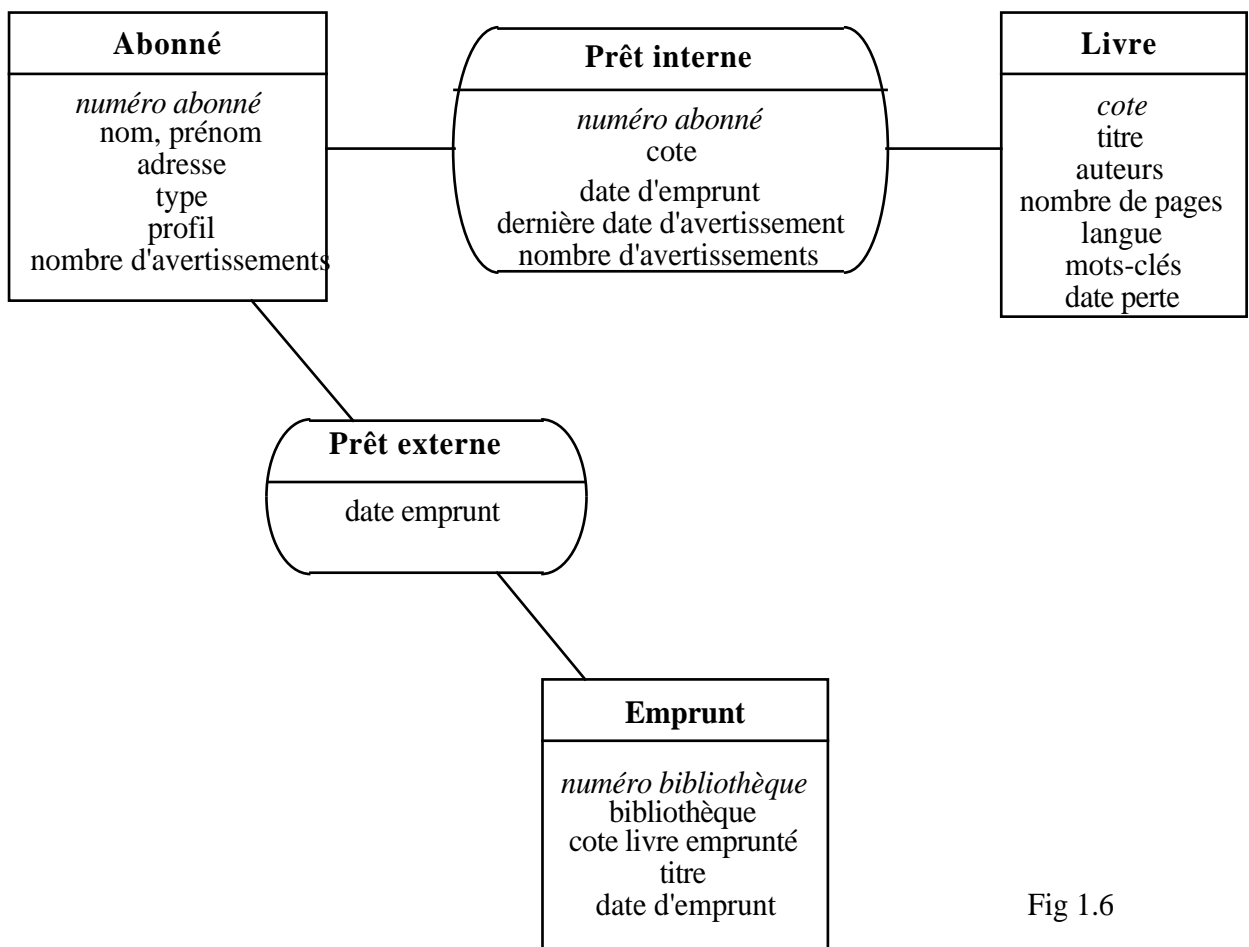


Fig 1.6

Choix des paramètres constants

Nous choisissons de mettre en oeuvre les paramètres constants minimum concernant l'aide à l'utilisateur; ce sont les possibilité *d'interrompre, de quitter, d'annuler et de transférer.*

En conséquence, tous les schémas qui suivent doivent être interprétés en fonction de ce choix.

Nouveau système
Procédure minimale administratif

But : gestion des achats de livres

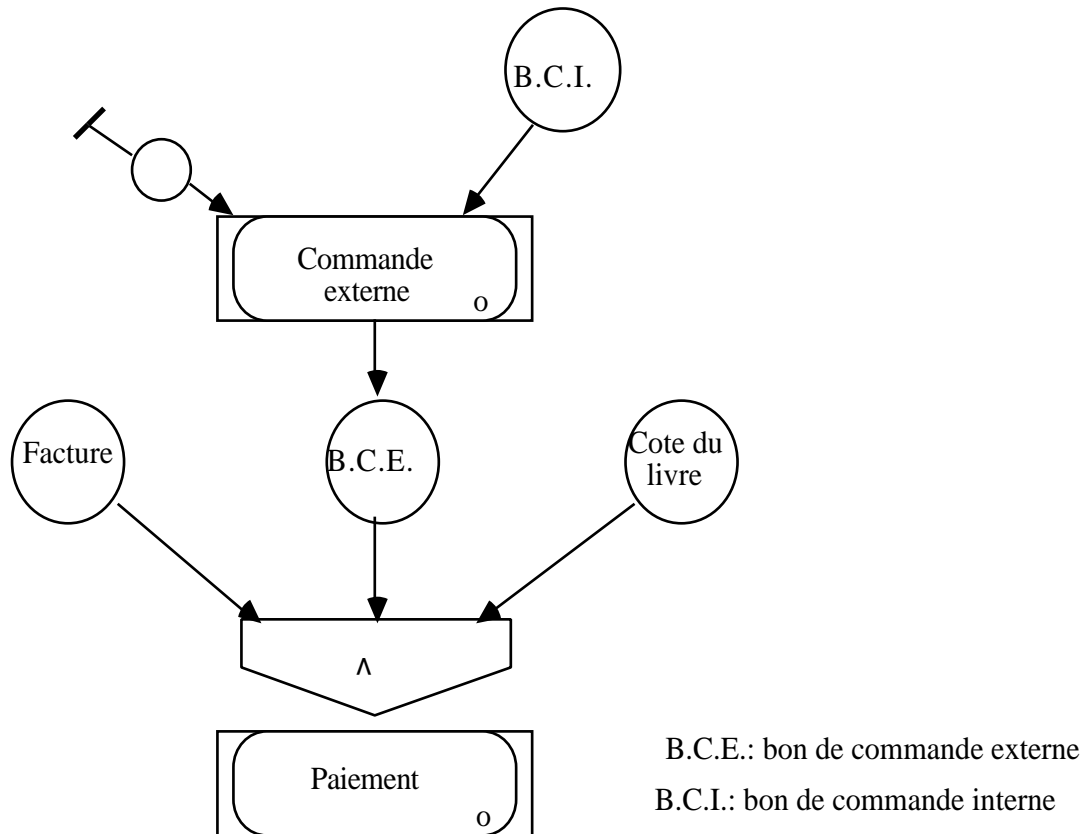


Fig 1.7

Nouveau système

But : emprunt de livres

Procédure minimale abonné

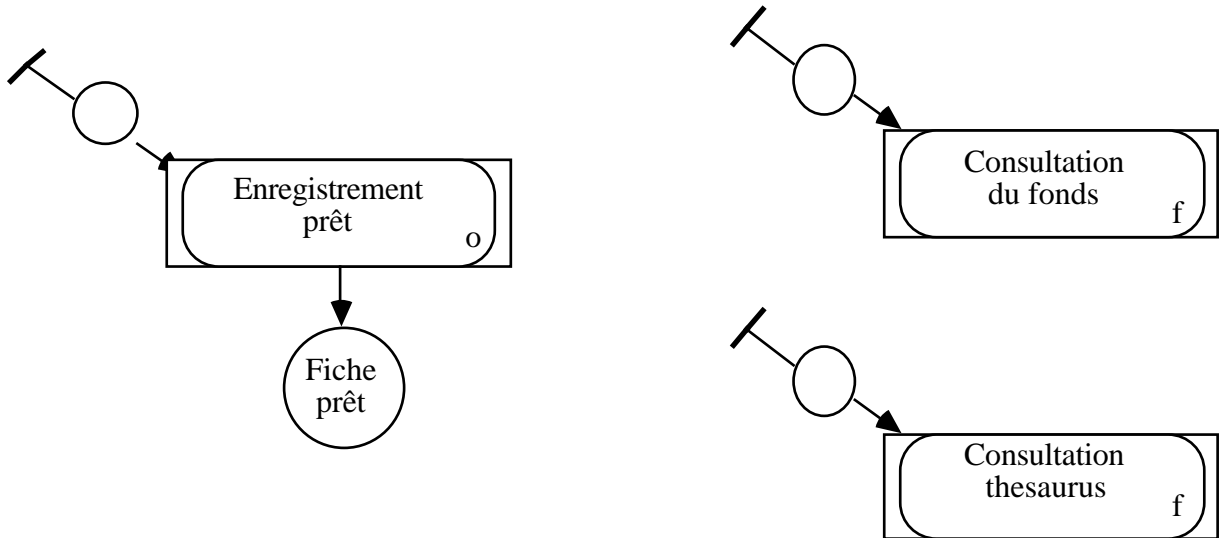


Fig 1.8

Procédure prévue abonné

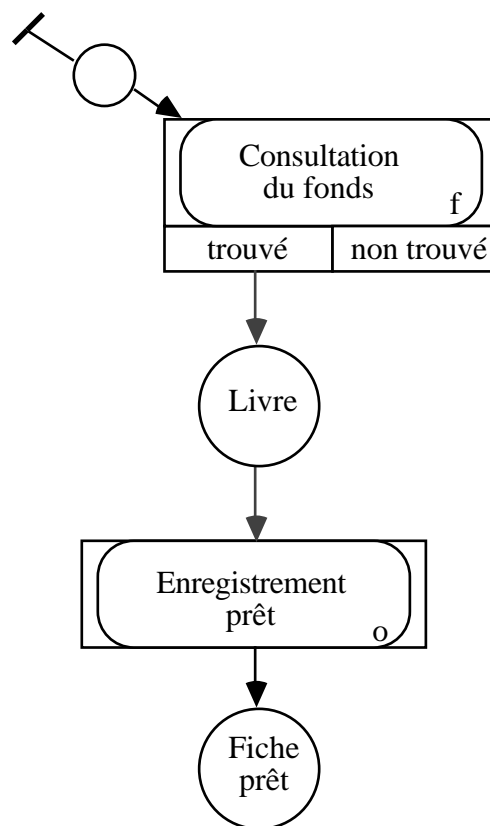


Fig 1.9

Nouveau système
Procédure minimale documentaliste

But 1: gestion du fonds propre

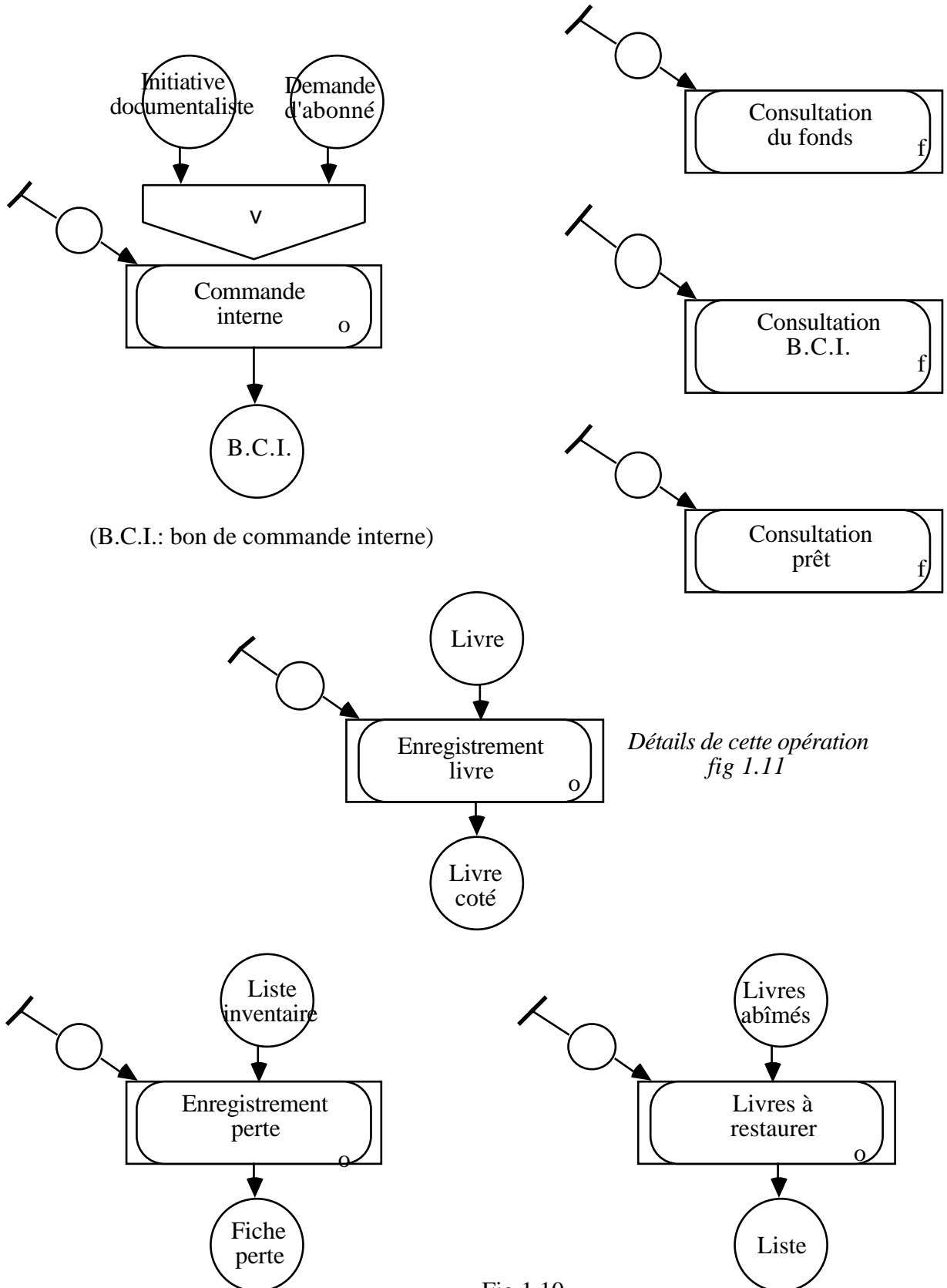


Fig 1.10

Remarques sur la figure 1.9

La précedence entre les opérations "Consultation du fonds" et "Enregistrement prêt" est une précedence facultative (indiquée en pointillé) car elle ne figure pas dans la procédure minimale de l'abonné.

Ici, elle engendre un enchaînement automatique car il n'y a pas de déclenchement par l'utilisateur de l'opération "Enregistrement prêt".

Cette précedence modifie l'opération "Enregistrement prêt" en ce sens qu'il y a un transfert automatique des données décrivant le livre consulté, ce qui minimise la saisie de la part de l'abonné.

L'opération "Enregistrement prêt" de la procédure abonné diffère de celle du même nom de la procédure documentaliste par le fait qu'il y a émission d'une fiche prêt qui sert à retirer le livre de la bibliothèque.

Remarques sur la figure 1.10:

Il n'y a pas de précedence permanente entre les opérations "Commande interne" et "Enregistrement" car il y a des livres qui parviennent à la bibliothèque sans avoir été commandés.

Les opérations "Consultation B.C.I" et "Consultation prêt" ont été rajoutées (par rapport à la description de la fonction1) car elles peuvent être utiles à la prise de décision.

Détail de l'opération "Enregistrement livre"

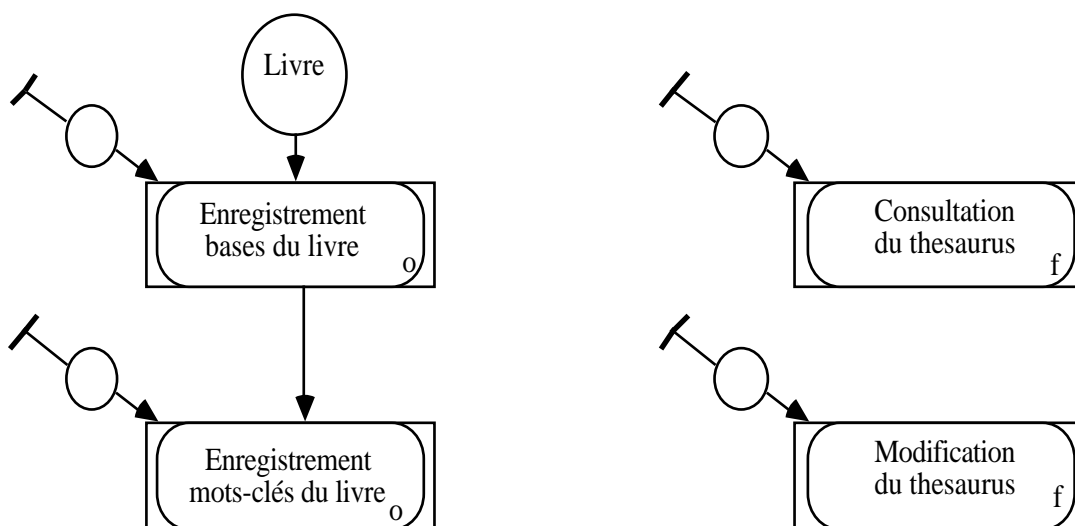


Fig 1.11

Procédure minimale documentaliste (suite)

But 2 : gestion des prêts

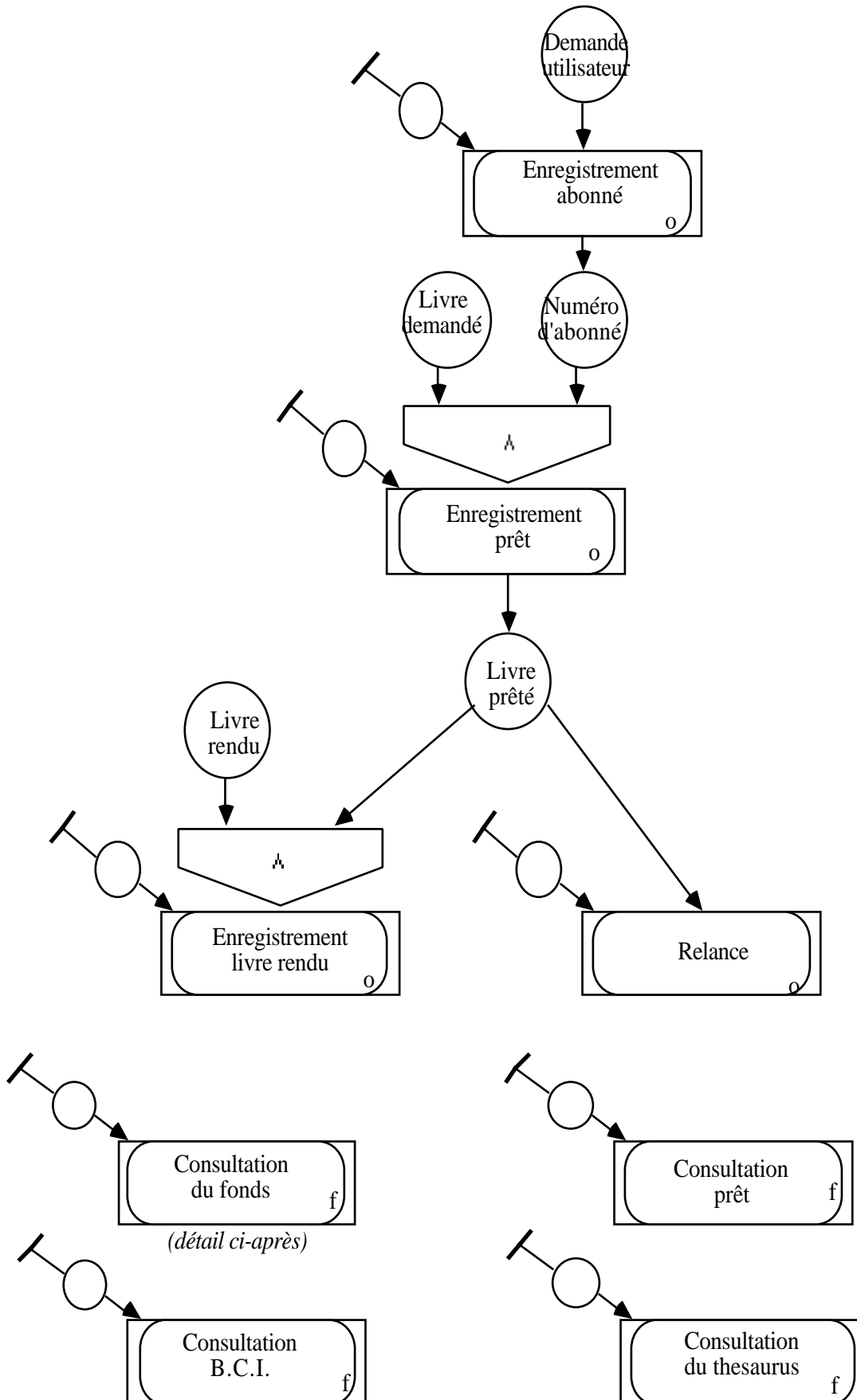


Fig 1.12

Détail de l'opération "Consultation du fonds"

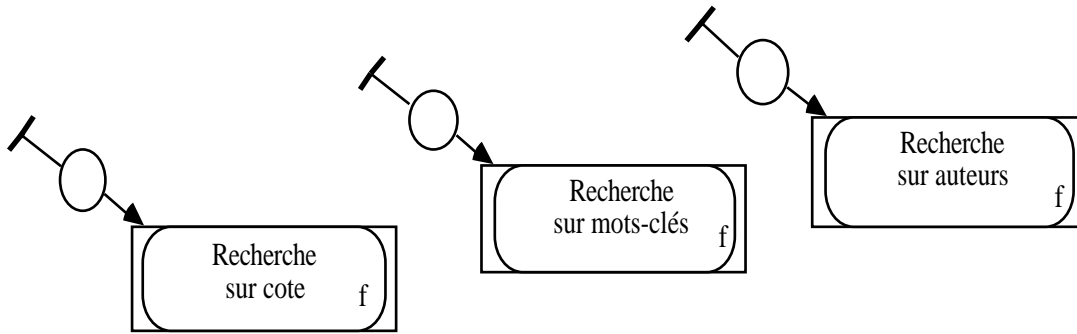


Fig 1.13

Procédure minimale documentaliste(suite)

But 3 : gestion des échanges avec d'autres bibliothèques

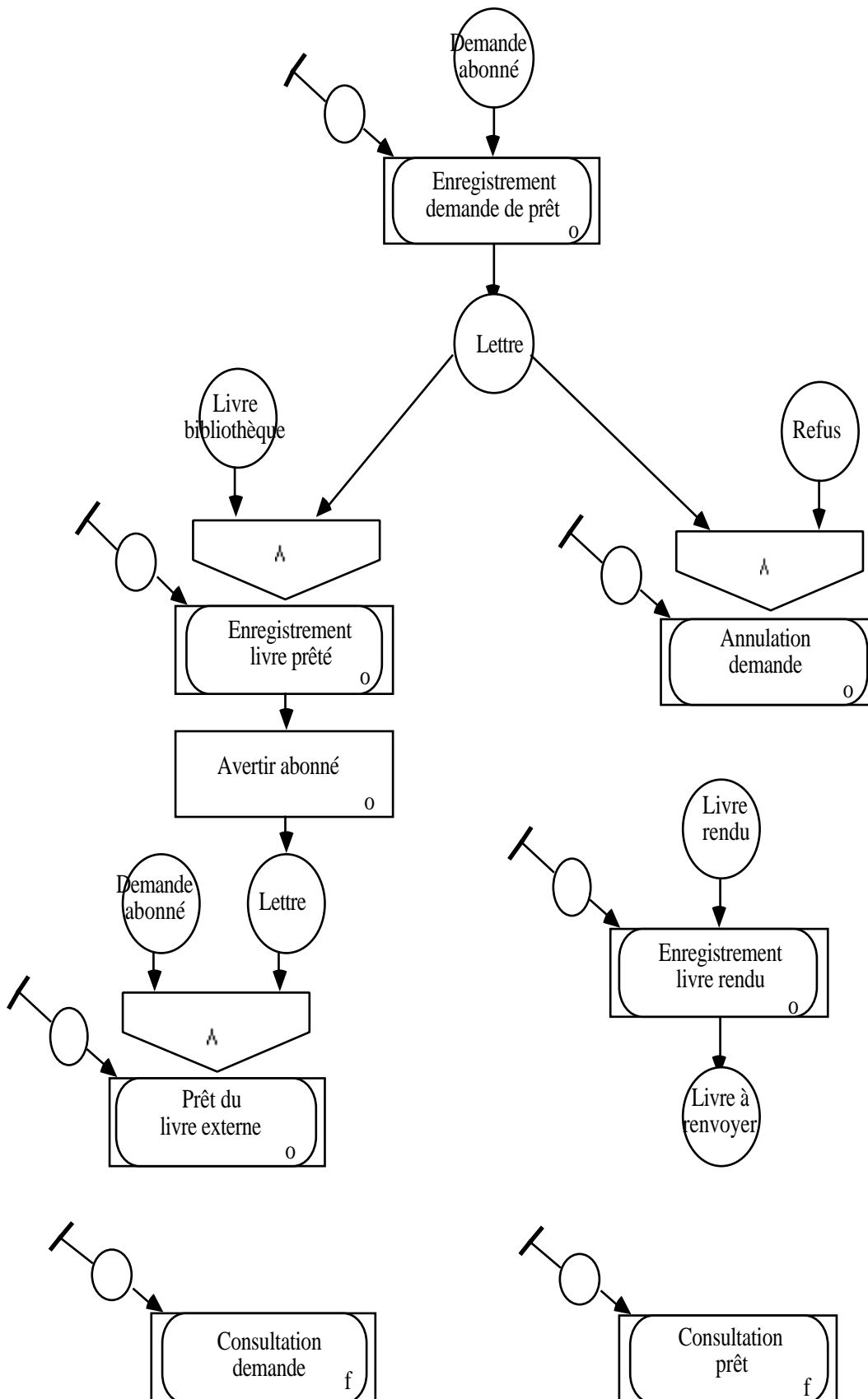


Fig 1.14

Nouveau système
Procédure effective documentaliste

But 1: gestion du fonds propre

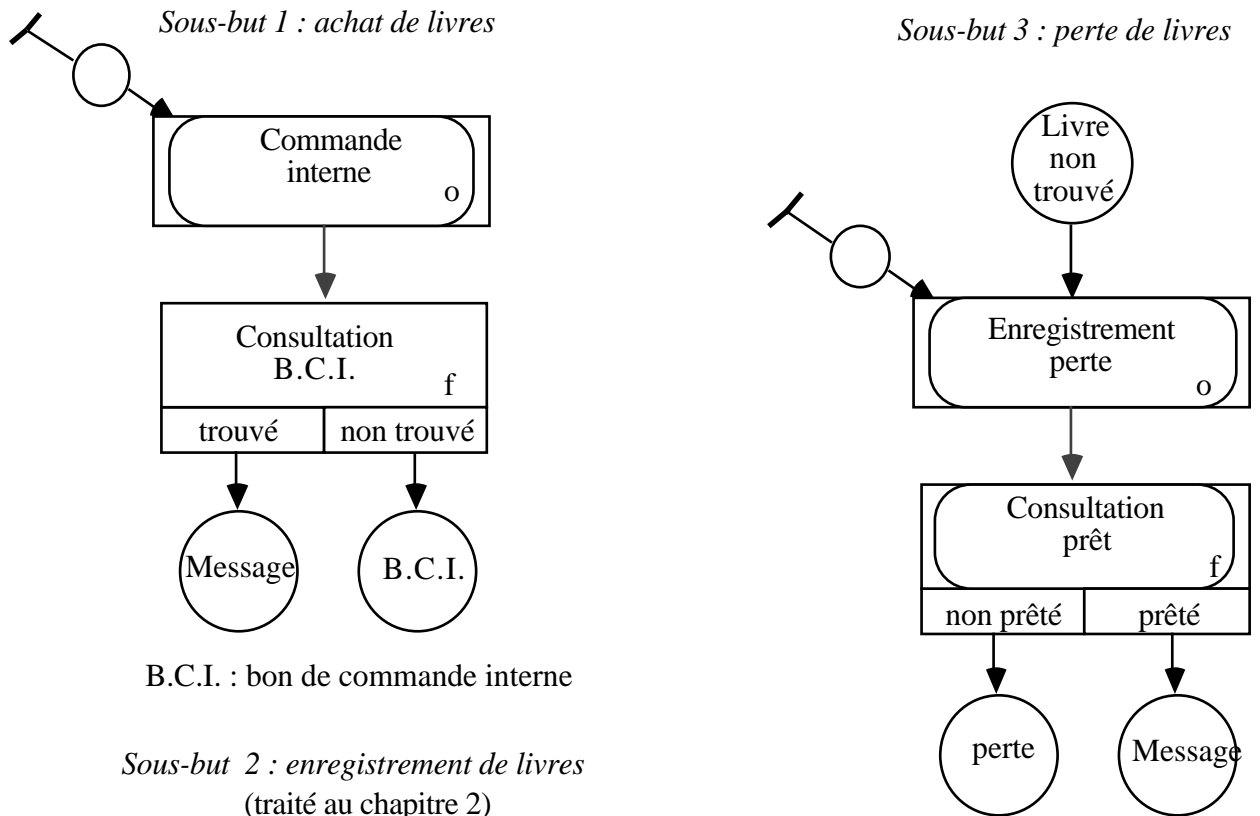


Fig 1.15

Remarques sur la figure 1.15

Pour minimiser le travail du documentaliste, l'opération "Consultation B.C.I" devient automatique quand le documentaliste déclenche l'opération "Commande interne".

De même, l'opération "Consultation prêt" est exécutée automatiquement à la suite de l'activation de l'opération "Enregistrement perte".

Nouveau système

Procédure effective documentaliste

But 2 : gestion des prêts

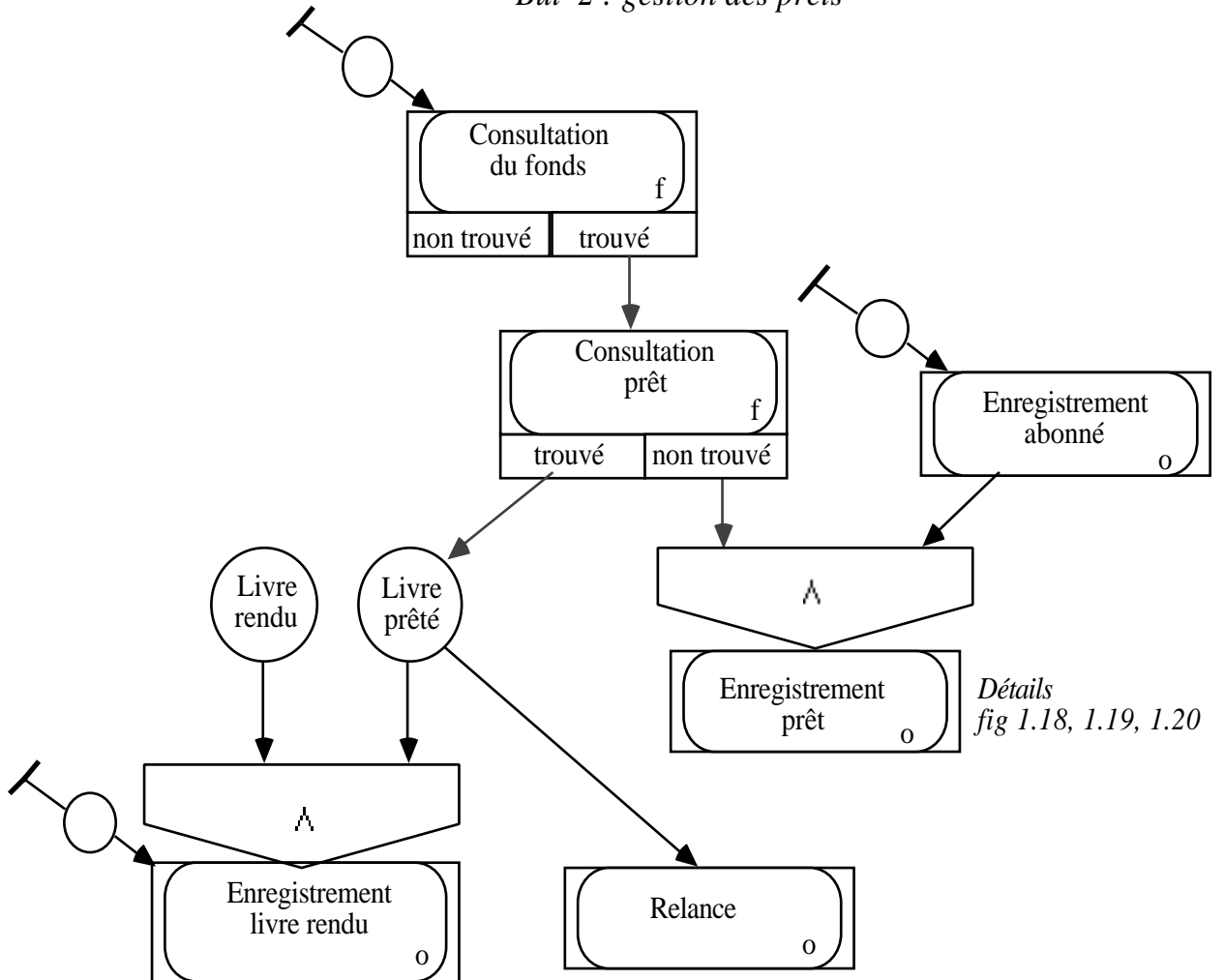


Fig 1.16

Représentation conceptuelle de l'opération "Enregistrement prêt"

Procédure prévue

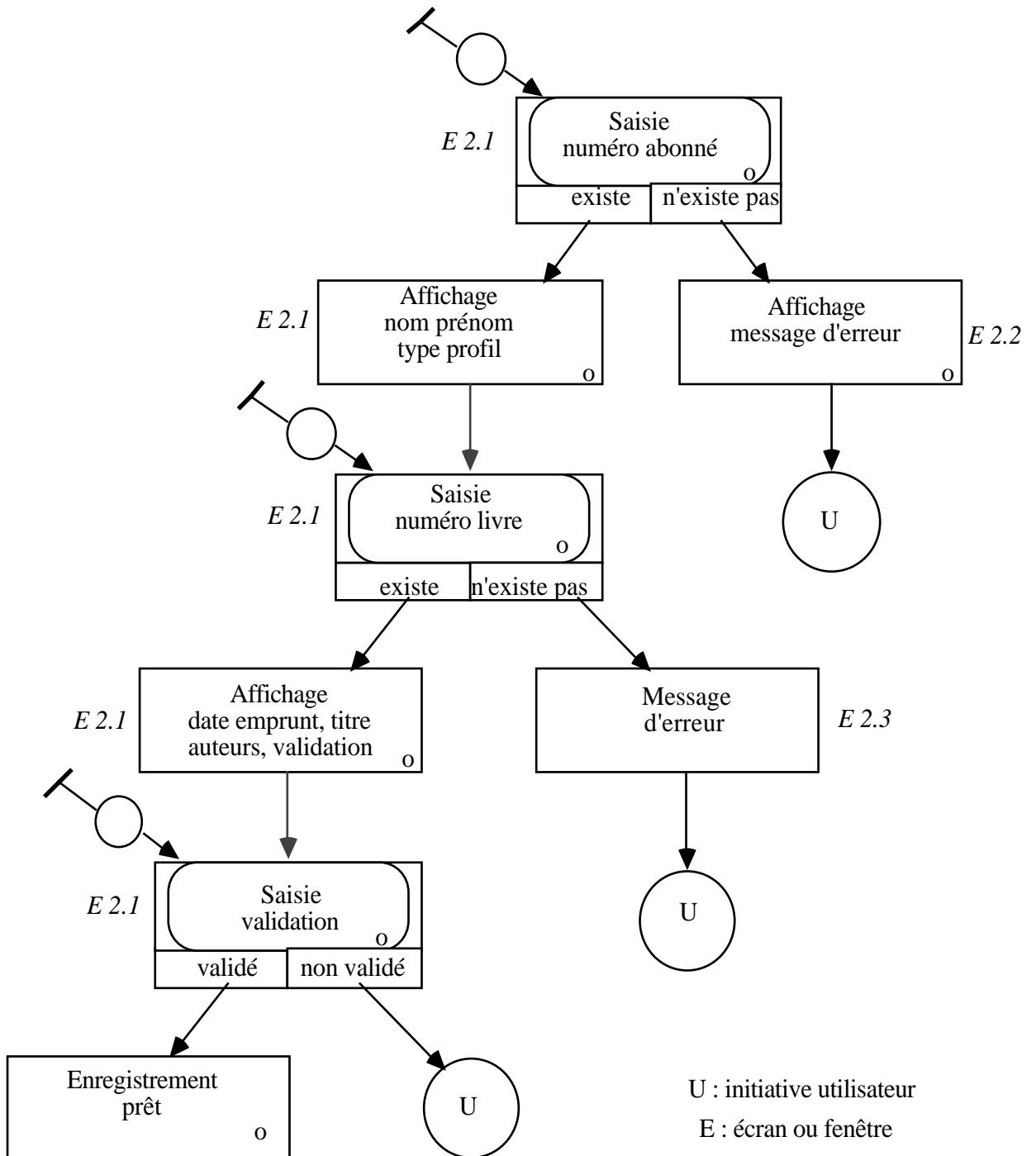


Fig 1.17

Remarques sur la figure 1.17

Cette figure décrit en détail l'opération "Enregistrement prêt" en descendant au niveau de détail le plus élevé que l'on peut atteindre avec ce formalisme (saisie d'une rubrique par l'utilisateur).

Le symbole "U" (initiative utilisateur) signifie que l'on passe le contrôle de la procédure à l'utilisateur qui peut alors activer soit une des commandes standard à sa disposition (remonter, quitter, annuler,...), soit une commande de la procédure activable à ce moment là.

L'utilisation de ce symbole a l'avantage de simplifier le schéma puisqu'il n'oblige plus à dire tout ce qui est possible ; par contre, il peut être complété en commentaires par la liste de ce qui est interdit, c'est-à-dire les opérations non activables à un moment donné.

La liste des commandes activables par l'utilisateur à un moment donné doit être visible pour l'utilisateur dans le menu (paramètres définis dans la Représentation Externe).

Un enchaînement automatique (pas de déclenchement utilisateur) se traduit à ce niveau de détail par un positionnement automatique du curseur sur l'écran.

Si on voulait le faire figurer dans le schéma de description de cette opération, il faudrait supprimer les déclenchements de l'utilisateur des opérations "Saisie numéro abonné", "Saisie numéro livre", "Saisie validation".

5.4.2.2 Conception de la Représentation Externe

Nous présentons une description de l'écran standard dans les figures 1.18 et 1.19 puis la Représentation Externe de l'opération "enregistrement prêt" (décrite ci-dessus) dans les figures 1.20 et 1.21.

Ecran standard

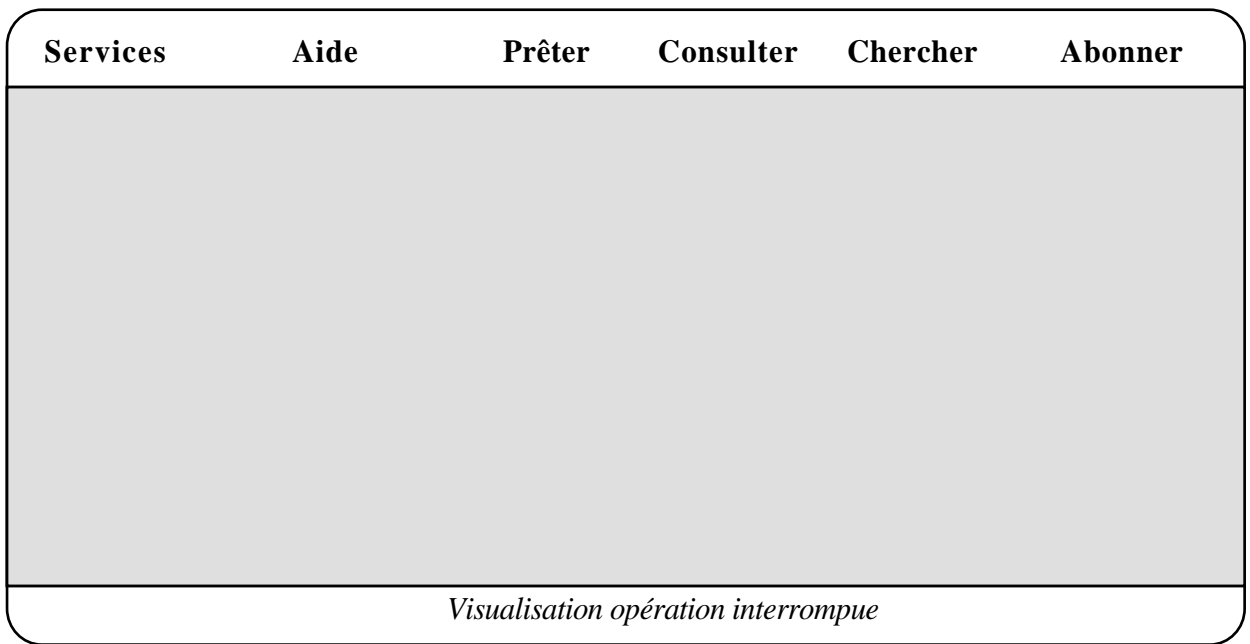


Fig 1.18

Visualisation du contenu des menus déroulants

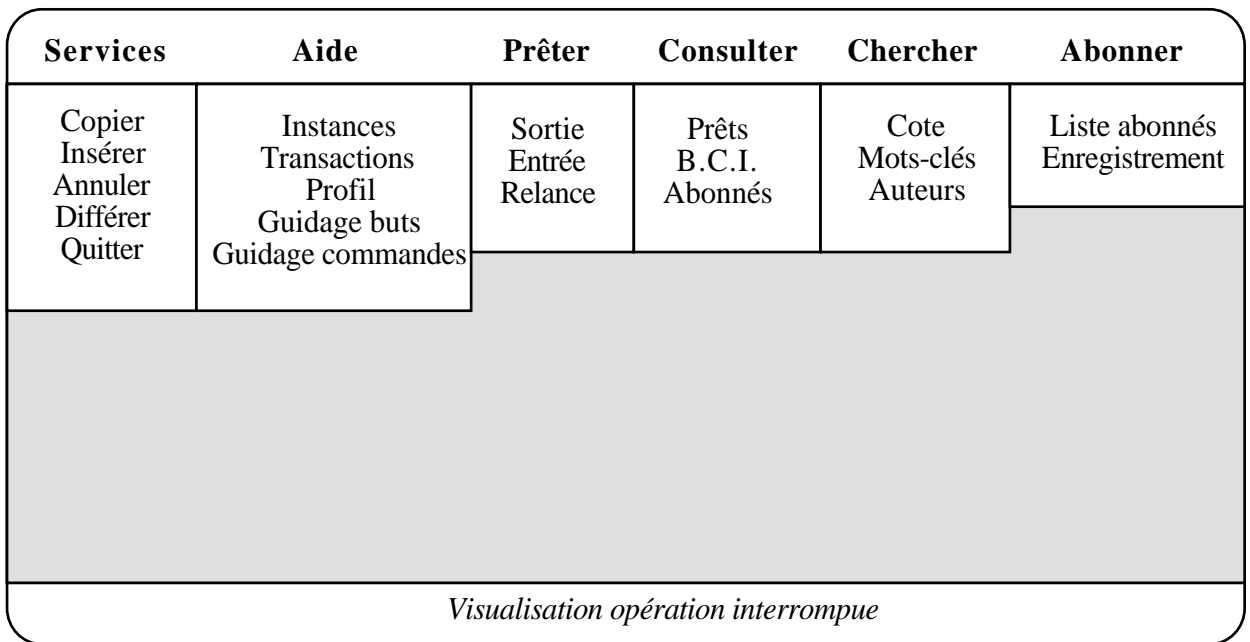


Fig 1.19

Ecran E 2.1

Services	Aide	Prêter	Consulter	Chercher	Abonner
Abonné					
Numéro	<input type="text"/>	Nom	*****		
		Adresse	*****		
		Type	**		
		Profil	**		
Livre emprunté					
Cote	<input type="text"/>	Titre	*****		
Date d'emprunt	*****	Auteurs	*****		
					Validation <input type="checkbox"/>
<i>Visualisation opération interrompue</i>					

Complété par l'utilisateur


* Complété par la machine

Fig 1.20

Remarques sur la figure 1.20


Dans la description de l'écran, nous distinguons visuellement les données qui sont saisies par l'utilisateur des données affichées par l'ordinateur.

Fenêtre E 2.2



Abonné inexistant !

Fenêtre E 2.3



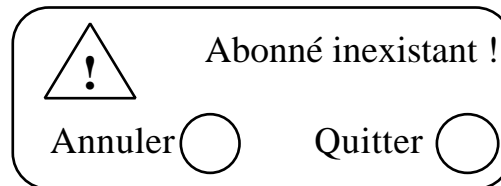
Cote inconnue !

Fig 1.21

Remarques sur la figure 1.21

Le terme LU signifie que l'utilisateur doit "cliquer" sur ce terme pour indiquer qu'il a lu le message d'erreur.

si il n'y avait qu'un choix possible pour l'utilisateur après une erreur, il serait plus judicieux de l'indiquer directement dans la fenêtre contenant le message d'erreur comme nous l'indiquons ci-dessous (l'utilisateur doit "cliquer" sur les "boutons" permettant de sélectionner les commandes ANNULER ou QUITTER).



Chapitre 6 :

DIANE

METHODE DE DIAGNOSTIC

Ce chapitre s'adresse plus particulièrement aux personnes qui se situent à l'extérieur du processus de conception, tel que nous venons de le décrire au chapitre 5, mais qui peuvent intervenir sur celui-ci en particulier pour améliorer la communication homme-machine. Il peut s'agir du correspondant informatique, d'un ergonome, d'un conseiller extérieur à l'entreprise ou d'un utilisateur.

Ce chapitre comprend trois parties . Dans la première partie, nous indiquons comment procéder aux tests d'un logiciel existant; dans la deuxième partie comment faire un cahier des charges permettant de spécifier les caractéristiques du logiciel amélioré et dans la troisième partie, nous exposons une étude de cas portant sur le diagnostic d'un logiciel existant.

6.1 Diagnostic d'un logiciel interactif

6.1.1 Démarche

Pour faire le diagnostic d'un logiciel existant, on dispose généralement de deux sources d'informations: d'une part les dossiers d'analyse et de programmation, d'autre part le logiciel lui-même tel qu'il a été réalisé.

Les dossiers d'analyse et de programmation correspondent respectivement à la Représentation Conceptuelle et à la Représentation Interne. Le logiciel réalisé correspond essentiellement à la Représentation Externe, même si on peut également y voir des éléments de la Représentation Conceptuelle traduits en Représentation Externe et un élément de la Représentation Interne, le temps de réponse.

Le dossier d'analyse peut fournir des indications intéressantes concernant le schéma de la base de données, la hiérarchie du menu, l'enchaînement des opérations et, éventuellement, les dessins

d'écran; le dossier de programmation indiquera les types de contrôle d'erreurs effectués et le détail des entrées/sorties.

Mais le diagnostic ne peut en aucun cas être réalisé uniquement au vu des dossiers d'analyse et éventuellement de programmation.

En effet, ces dossiers sont rarement complets car de nombreux détails se règlent souvent au stade de l'écriture des programmes; de plus, ils peuvent être erronés par le fait que des options prises au niveau de l'analyse n'ont pas été respectées au niveau de la réalisation.

Aussi, dans tous les cas, les éléments pertinents pour la communication homme/machine que nous relevons dans ces dossiers devront être testés sur le logiciel réalisé.

Le diagnostic est possible même si nous ne disposons que du logiciel réalisé car il est possible de recomposer les éléments des dossiers d'analyse et de programmation qui nous intéressent en examinant le logiciel réalisé même si cette procédure peut s'avérer plus longue.

L'essentiel de notre exposé porte sur l'examen du logiciel car, comme nous venons de le voir, c'est un passage obligé pour notre étude.

La méthode la plus efficace consiste à procéder en deux étapes:

- tout d'abord, on teste soi-même le logiciel sur les critères que nous décrivons dans la grille d'analyse ci-après; ces tests permettent de dégager une grille d'observation ,
- ensuite, on effectue des observations sur des utilisateurs du logiciel à partir de la grille d'observation élaborée précédemment.

Dans les deux cas, on doit procéder à des recopies d'écran systématiques en cours de tests pour pouvoir recomposer ultérieurement les séquences suivies et s'assurer que l'ensemble du logiciel a été testé; c'est pour cela que la confrontation avec le dossier d'analyse peut s'avérer fructueuse.

6.1.2 Grille d'analyse

Cette grille d'analyse repose en grande partie sur les paramètres de l'interface vus par l'ergonome (voir chapitre trois) ; cette approche permet effectivement de tester les possibilités de communication avec un logiciel d'un point de vue extérieur à sa conception.

Pour chaque écran , on observe les paramètres suivants:

- *vocabulaire*

La première question qui se pose est de savoir si le vocabulaire utilisé, que ce soit pour nommer les commandes et les données ou

pour les messages de services ou d'erreurs est ambigu pour l'utilisateur; ces ambiguïtés se traduisent, soit par des erreurs systématique des utilisateurs, soit par une sous-utilisation des possibilités du logiciel. Dans tous les cas, l'utilisateur a des difficultés à mémoriser ce vocabulaire.

La deuxième question concerne les abréviations qui, si elles existent, doivent être mnémoniques: l'utilisateur doit pouvoir trouver une règle simple lui permettant de passer du mot complet à son abréviation; là aussi, l'homogénéité de l'application de cette règle est essentielle.

- *syntaxe*

Le premier élément à tester est l'homogénéité: des actions identiques de la part de l'utilisateur (quand elles sont autorisées) ont des effets identiques sur le programme; inversement, une même action de l'ordinateur est toujours activée par la même commande; ceci doit être vérifié sur l'ensemble des écrans.

Le deuxième élément est la simplicité du point de vue de l'utilisateur.

Si ces deux éléments ne sont pas respectés, l'utilisateur ne pourra pas développer d'automatismes, ce qui se traduira au mieux par une lenteur d'exécution et au pire par des erreurs de manipulation.

- *dispositifs d'entrée*

Les observations d'erreur liées strictement au dispositif d'entrée sont assez rares maintenant, car les dispositifs d'entrée en informatique des organisations sont peu nombreux (clavier, souris,...) et assez standardisés.

Par exemple, toutes les marques d'ordinateur proposent aujourd'hui des claviers AZERTY alors que le clavier QWERTY est, en France, une source continue d'erreurs pour des utilisateurs familiarisés aux normes françaises.

Par contre, d'autres éléments du clavier (touches-fonctions...) et les "souris" ne sont pas normalisés, ce qui peut être une source d'erreurs aisément observable.

- *dispositifs de sortie*

Nous ne parlons pas ici des caractéristiques matérielles des écrans (bien qu'elles jouent un rôle essentiel dans la fatigue visuelle) car elles ne relèvent pas de l'ergonomie du logiciel et donc de notre marge de manoeuvre concernant le logiciel.

La première chose à observer est la lisibilité générale qui se traduit par un écran qui n'est pas surchargé d'informations, avec des zones distinctes pour des objets informationnels distincts, de manière que

la structure de l'écran apparaisse clairement à la suite d'un parcours visuel rapide.

La deuxième chose à observer est l'homogénéité de la présentation entre les différents écrans de même type (écrans de menus, écrans de saisie, écrans d'explication,...). Par exemple, le nom de l'opération que l'on est en train d'exécuter doit toujours apparaître au même endroit de l'écran et avec le même graphisme.

Une présentation générale surchargée ou non homogène engendre lenteur de lecture et fatigue .

Puis, nous examinons plus en détail chaque zone de l'écran pour savoir si elles sont pertinentes ou non pour l'utilisateur; inversement, nous regardons si toutes les données pertinentes au niveau de cet écran y figurent.

La pertinence des zones peut être longue à tester et demande une connaissance approfondie du travail ou des observations.

- temps de réponse

L'observation des temps de réponse est simple et leur acceptabilité obéit à des règles simples développées au chapitre 3. Il faut vérifier que pour des temps de réponse supérieur à quatre secondes, un message d'information apparaît.

Dans le cas où le logiciel se déconnecte au bout d'un certain temps de non utilisation, il faut vérifier que ce laps de temps est compatible avec la nature de la tâche entreprise. Par exemple, une tâche de type décisionnel s'accompagne de délais de réponse plus long de la part de l'utilisateur.

- logique d'utilisation

Pour les écrans de menu, on peut vérifier si l'ensemble des commandes présenté sur un écran suit une logique d'utilisation c'est-à-dire si l'utilisateur y trouve un ensemble cohérent du point de vue de la réalisation de sa tâche.

Ce critère est repris à un niveau plus global avec l'enchaînement des écrans.

- traitement des erreurs

Le premier élément à tester est la capacité du logiciel à percevoir les erreurs de l'utilisateur. Pour cela, il faut dans un premier temps imaginer un ensemble d'erreurs possibles, puis compléter si nécessaire par des observations.

Le deuxième test porte sur l'intelligibilité du message d'erreur et sa pertinence par rapport à l'erreur commise.

Le troisième test porte sur les possibilités et les facilités offertes pour rectifier les erreurs.

L'ensemble des tests sur les erreurs constitue un travail important de la part de l'intervenant et des utilisateurs mais il est crucial pour la qualité du logiciel.

- *aide*

Il faut identifier les différents types d'aide offerts à l'utilisateur (explicitation de commandes, explicitation de procédures, inférences) et pour chaque type d'aide procéder aux vérifications suivantes.

La première vérification concerne la possibilité d'être guidé à tout moment de l'utilisation du logiciel.

La deuxième vérification porte sur la cohérence des explications avec les faits: il faut vérifier si les explications à caractère général sur les commandes correspondent bien à la réalité du comportement du logiciel dans tous les cas.

Pour les enchaînements d'écran, il faut tester les éléments suivants:

- *l'arborescence du menu*

Il faut vérifier que l'ensemble de l'arborescence du menu correspond bien à une logique d'utilisation, c'est-à-dire que les commandes regroupées à un certain niveau correspondent bien à un ensemble de commandes qui ont un rapport entre elles du point de vue du travail de l'utilisateur.

Si les enchaînements d'écran sont automatiques, il faut s'assurer que ce fait constitue une aide pour l'utilisateur et non pas une gêne; inversement, il faut essayer de détecter si certains enchaînements d'écran ne devraient pas devenir automatiques.

Si les interruptions d'opérations ne sont pas permises par le système, il faut vérifier que l'on peut naviguer aisément entre les opérations que l'on a besoin d'utiliser en parallèle comme, par exemple, "consulter" avant ou après "enregistrer".

Les tests sur ces éléments peuvent être simulés par l'intervenant, mais il est indispensable qu'ils soient réalisés avec des utilisateurs.

- *les commandes standard d'enchaînement*

Il faut vérifier que les commandes standard d'enchaînement entre écrans comme "sortie", "aide", etc..., sont disponibles à tout moment et produisent toujours les mêmes effets!

Si certaines commandes ne sont pas exécutables dans certaines parties du programme, elles doivent être signalées à l'utilisateur;

inversement, on doit indiquer les commandes standard disponibles à chaque moment.

- l'homogénéité entre écrans

Nous devons vérifier que la syntaxe, les dispositifs d'entrée et la présentation de l'écran (dispositif de sortie) sont homogènes sur l'ensemble du logiciel et même sur l'ensemble des logiciels manipulés par les différents postes de travail.

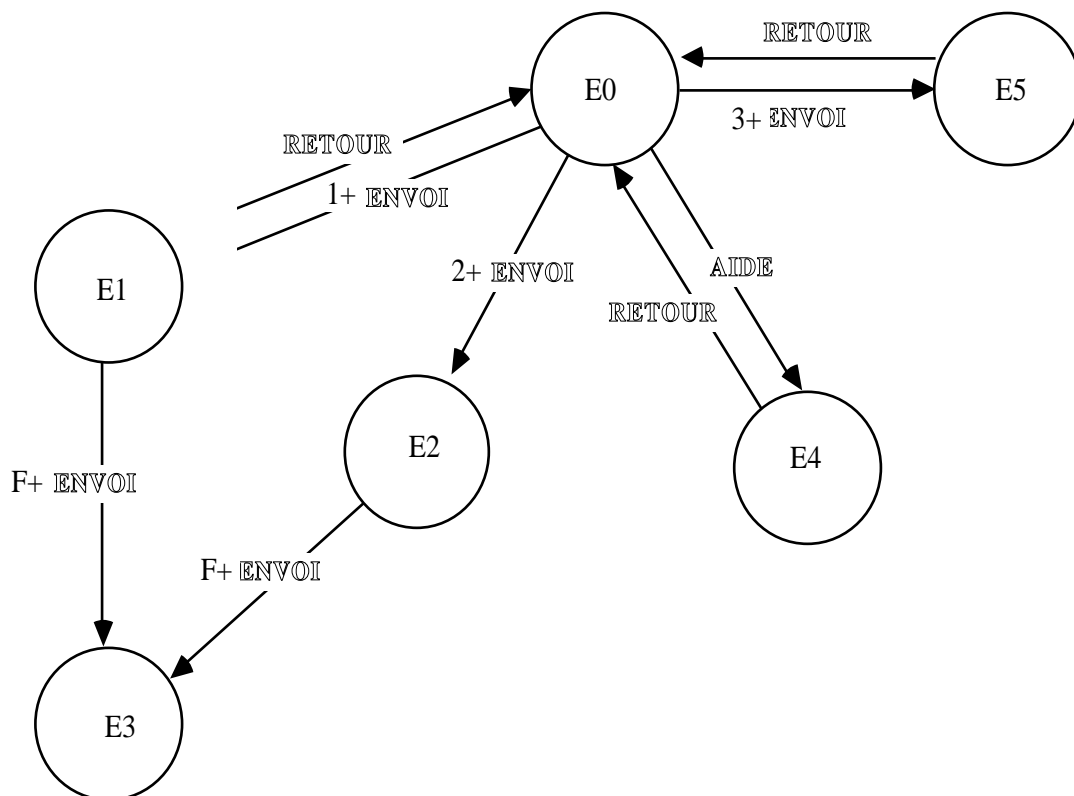
Pour permettre un diagnostic plus aisé sur l'enchaînement des écrans, on peut représenter ces enchaînements sous la forme d'un automate d'états finis comme le propose la méthode Use (Was, 82) avec les conventions suivantes:

- les états (visualisées par des cercles) représentent les écrans ou les fenêtres ou les lignes selon le mode d'interaction minimum entre l'homme et la machine.

- les transitions (visualisées par des flèches) représentent les actions de l'utilisateur permettant de basculer d'un état à l'autre.

Nous présentons ci-après un exemple de ce formalisme:

Exemple d'enchaînement d'écrans



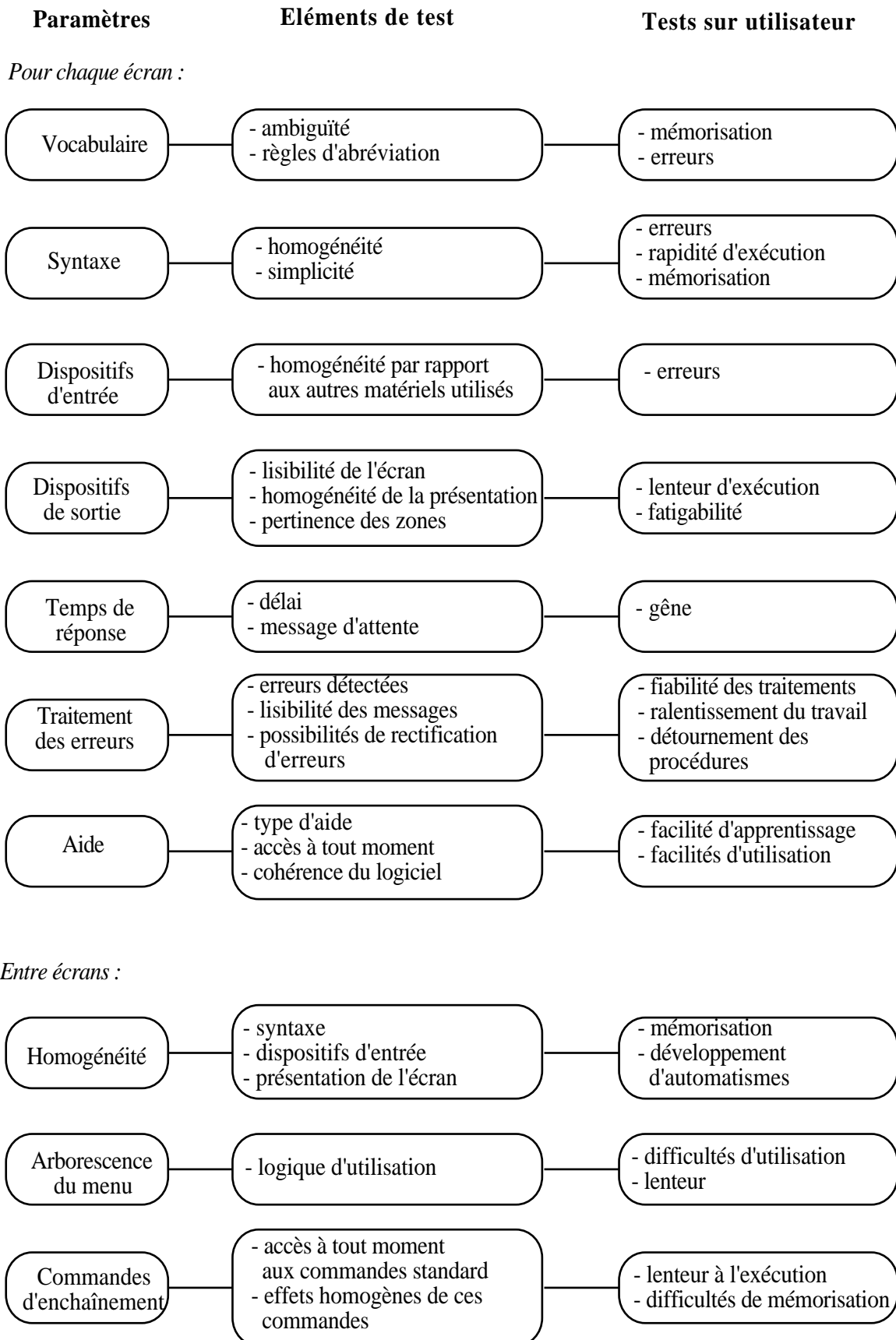
Si l'on veut que ce type de schémas reste lisible, il est la plupart du temps impossible de faire figurer sur un même schéma l'ensemble des transitions possibles.

Aussi, en pratique, on réalise un premier schéma "descendant" où on visualise les enchaînements en partant du premier écran vers tous les autres; puis on fait un ou plusieurs autres schémas pour représenter toutes les autres transitions possibles (quitter, aide,...).

Des exemples de ces schémas sont donnés dans l'étude de cas RTR présenté dans la troisième partie de ce chapitre.

Dans le tableau de synthèse ci-dessous, nous rappelons l'ensemble des éléments à tester et pour chacun d'eux nous indiquons quels types de tests doivent être pratiqués avec les utilisateurs en plus d'une expérimentation directe par l'intervenant.

Grille d'analyse d'un logiciel (synthèse)



6.2 Cahier des charges d'un logiciel interactif

Dans cette étape, nous présentons un canevas de cahier des charges du logiciel à destination des informaticiens responsables de la réalisation du logiciel.

L'objectif de ce cahier des charges est de traduire les observations que l'on vient de faire sur le logiciel en spécifications opérationnelles pour les informaticiens.

On peut, à ce sujet, remarquer qu'il est possible de faire un cahier des charges pour un logiciel interactif uniquement à partir de la Représentation Externe, c'est-à-dire en partant du point de vue de l'utilisateur; les informaticiens se chargeant alors d'élaborer une Représentation Conceptuelle et une Représentation Interne compatibles avec cette Représentation Externe.

Ce cahier des charges se présente comme un cahier des charges classique de logiciel informatique auquel on rajoute un ensemble de spécifications propres à l'interaction homme-machine.

C'est ce dernier aspect que nous développons ci-dessous.

Nous rappelons qu'un cahier des charges classique comprend l'ensemble des spécifications concernant les données (rubriques, accès, volumes,...) et les traitements (algorithmes, sécurité, confidentialité,...).

Nous rajoutons à ces spécifications celles concernant les Représentations Externes (ou vues externes) des différents postes de travail concernés par le logiciel. Il est tout à fait inutile de donner des spécifications concernant la Représentation Interne car cela relève de la compétence et de la responsabilité de l'informaticien qui doit concevoir cette Représentation Interne de manière à ce qu'elle puisse satisfaire les spécifications de la Représentation Externe.

On ne donne pas non plus de spécifications directes concernant la Représentation Conceptuelle mais la Représentation Externe contient des paramètres provenant de la Représentation Conceptuelle comme les enchaînements d'écrans ou la répartition du pilotage entre l'homme et la machine.

Les spécifications concernant la Représentation Externe se subdivisent en deux sous-ensembles; le premier concerne les spécifications à caractère général, le deuxième les spécifications propres à l'application et à chaque poste de travail.

6.2.1 Spécifications générales

L'objectif de ces spécifications est de fournir un cadre général de réalisation du logiciel permettant de créer une homogénéité maximale à l'intérieur du logiciel et avec les logiciels existants s'il y a lieu.

Ces spécifications doivent porter sur les paramètres suivants:

- les commandes standard qui doivent être actives (sauf exceptions) sur l'ensemble du logiciel. Il s'agit des commandes d'aide à l'utilisation que nous avons présentées dans les chapitres 2 et 3; les commandes les plus usuelles concernent les possibilités de quitter, d'annuler, de revenir en arrière, d'interrompre. Il vaut mieux spécifier également le vocabulaire de désignation de ces commandes et les abréviations, surtout s'il existe déjà des logiciels les utilisant, afin que ce vocabulaire reste identique et n'entraîne pas d'ambiguïté sémantique chez les utilisateurs.

- la présentation générale des écrans qui doit être identique sur l'ensemble du logiciel et de préférence compatible avec les logiciels existants. Nous entendons par présentation générale des écrans les choix concernant l'emplacement des zones de menu, de saisie de données, de messages d'erreur ainsi que la forme visuelle de ces paramètres; dans la forme visuelle, nous décrivons le type de caractère (gras, italique, surbrillance, couleur,...), la forme des fenêtres ou des zones, l'impression du fond des zones (motifs, couleurs,...), les conventions de représentations (touches-fonctions,...).

Le plus simple est de présenter une ou des images d'écran ou de fenêtres (selon que l'on a un logiciel de mono ou de multifenêtrage) avec une représentation des différentes formes visuelles.

- les dispositifs d'entrée qui doivent être également homogènes avec les autres matériels et logiciels existants; ces spécifications portent sur le type de clavier (AZERTY, accentué, mathématique...) et sur le mode de désignation des commandes (frappe au clavier, touche-fonction, souris,...).

- la syntaxe du langage de commande comme la validation des écrans ou l'indication de fin de saisie des données qui doivent aussi être homogènes.

Il faut bien se rendre compte que tout ce qui n'aura pas été spécifié ici a plus de chance d'être hétérogène qu'homogène surtout si la réalisation (ce qui est le cas le plus courant) est confiée à plusieurs

informaticiens. Pour s'en convaincre, il suffit de se reporter à l'étude de cas RTR décrit dans le paragraphe suivant et qui ne constitue pas un cas d'exception.

6.2.2 Spécifications particulières à l'application

Il faut d'abord définir les différents types de poste de travail concernés par l'application dans la mesure où ces types de postes correspondent à des vues externes différentes de la même application (par exemple les postes correspondant à différents niveaux de responsabilités).

Puis, pour chaque poste de travail, on définit la ou les Représentations Externes; on peut avoir plusieurs Représentations Externes pour un poste de travail (correspond aux différents types de procédures définies au chapitre 2) si on a plusieurs types d'utilisateurs (expérimentés, novices) ou des utilisateurs ayant des procédures effectives différentes.

Pour chaque Représentation Externe, nous spécifions les paramètres suivants:

- la répartition du pilotage entre l'homme et la machine qui comporte trois aspects; la spécification des enchaînements entre écrans, les écrans et données obligatoires ou facultatives à saisir et les déclenchements d'écrans à l'initiative de l'utilisateur ou de l'ordinateur.
- l'ensemble des commandes et la hiérarchie de ces commandes en menu suivant une logique d'utilisation; les noms de commandes doivent être spécifiés soit quand ils correspondent à un vocabulaire spécialisé, soit pour être compatibles avec des commandes de logiciels existants.
- les erreurs indispensables à détecter et le contenu des messages d'erreur correspondants; on peut rajouter éventuellement les possibilités de correction d'erreurs (retour-arrière, défaire,...).
- la présentation des écrans de saisie qui doivent correspondre le plus possible à des présentations déjà existantes comme les formulaires ou les écrans.
- le vocabulaire des spécialistes afin qu'il soit utilisé dans la dénomination des données et éventuellement des commandes.

6.3 Etude de cas " RTR "

Le cas RTR (Réseau de Transport par Rail) est inspiré d'un cas réel utilisé dans le cadre de la télématique grand public.

Ce cas a pour but d'illustrer la méthode permettant de faire un diagnostic d'un logiciel interactif.

Il s'agit, en l'occurrence, d'un logiciel télématique destiné au grand public, ce qui facilite le diagnostic en ce sens que nous n'avons pas besoin de faire des tests sur des types d'utilisateurs différents (expérimentés, novices,...) et que la notion de procédure effective n'est plus utile.

Pour ce type de logiciel, une première expérimentation, directement effectuée par l'analyste en fonction de la grille d'analyse définie précédemment, permet de faire un certain nombre de remarques utiles à la préparation de tests systématiques auprès d'un échantillon d'utilisateurs.

Nous présentons ci-dessous quelques-unes des images d'écrans sur lequel nous pouvons a priori formuler un certain nombre de critiques, compte-tenu des éléments ergonomiques présentés précédemment. Ces constatations ne sont pas forcément exhaustives car elles n'ont pas été validées sur un nombre suffisant d'utilisateurs.

Nous présentons ensuite des remarques ayant trait aux enchaînements entre les écrans.

6.3.1 Analyse des écrans

La figure 2.1 reproduit le premier écran du logiciel RTR sur lequel nous pouvons faire les remarques suivantes:

- la première remarque a trait au fait que la syntaxe n'est pas homogène; en effet, cet écran propose un choix entre quatre possibilités ; pour obtenir trois d'entre elles, on doit taper une abréviation puis appuyer sur la touche "ENVOI", alors que pour la quatrième, il faut appuyer sur la touche "SUITE". Comme nous le verrons, cette hétérogénéité se reproduit tout au long du logiciel; la conséquence en est que l'utilisateur ne peut pas acquérir d'automatismes concernant l'envoi d'un message à l'ordinateur; cela nécessite plus de concentration et occasionne des erreurs.

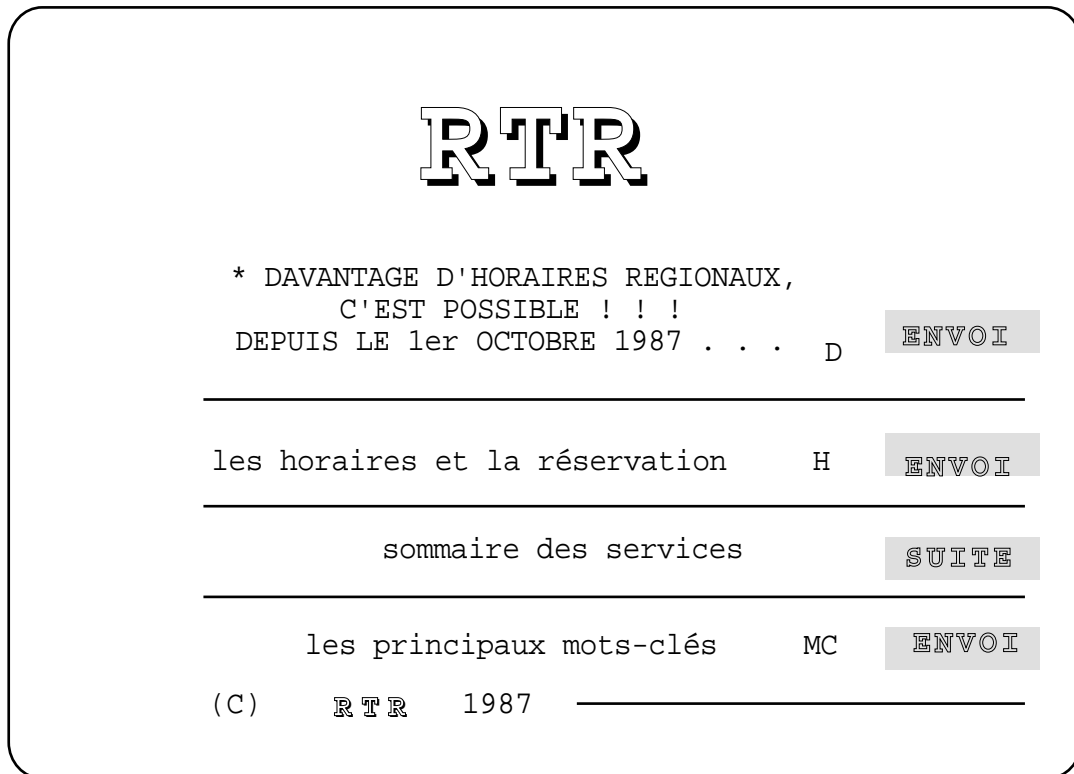


Fig 2.1

- la deuxième remarque est relative à l'ambiguïté des messages concernant le premier et le dernier choix: l'utilisateur n'a qu'une idée vague ou pas d'idée du tout sur ce que peut apporter la consultation de "Davantage d'horaires régionaux, c'est possible!!!" et "les principaux mots-clés". Par contre les deux autres possibilités sont explicites.

- la troisième remarque porte sur la non-pertinence d'une zone de l'écran; à la dernière ligne, nous avons le message "(C) RTR 1987" qui n'est d'aucune utilité pour l'utilisateur car il sait déjà par la première ligne qu'il est en train de consulter RTR, on peut penser qu'il sait en quelle année il est et la signification du (C) n'est pas nécessaire pour utiliser le logiciel. Cette zone inutile, du point de vue de l'utilisateur, se retrouve avec des variantes de présentation sur tous les écrans.

La figure 2.2 est obtenue en appuyant sur la touche "SUITE" à partir du premier écran; on peut l'obtenir aussi à d'autres moments en appuyant sur la touche "SOMMAIRE"; ceci est une autre forme de l'hétérogénéité de la syntaxe.

Nous avons choisi cet écran car il est très caractéristique des abréviations non mnémoniques en ce sens que ces abréviations ne sont pas constituées à partir d'une règle unique facilement mémorisable.

Les abréviations des commandes 2,3,4,7,9,11,12 reprennent des lettres contenues dans le vocabulaire de ces commandes et peuvent donc être plus facilement mémorisées (une étude plus fine s'imposerait); mais cette règle n'est pas appliquée pour les autres commandes.

RTR		SOMMAIRE
INFORMATIONS VOYAGEURS		
1	générales et régionales	INFV
2	informations personnalisées	PERS
3	les horaires	HORA
4	la réservation des places	RESA
INFORMATIONS MARCHANDISES		
5	colis, lots, déménagements	SERNAM
6	transports par wagons complets	MARC
7	transports en conteneurs	CONT
8	CONNAITRE LA RTR	INFO
9	l'actualité AC / 10 panorama	SE
11	les principaux mots-clés	MC
12	aide à l'utilisation du minitel	AI
tapez le N° ou le MOT choisi et		ENVOI
RTR	(C) 1987	

Fig 2.2

La commande 1 a une abréviation correspondant à la ligne précédente qui est un libellé général.

La commande 5 a comme abréviation le nom d'une entreprise!

Les commandes 6 et 10 ont une abréviation mystérieuse pour un utilisateur ordinaire.

La commande 8 a une abréviation proche de celle de la commande 1 et éloignée de la commande qu'elle désigne.

Nous pouvons faire également deux remarques sur la présentation:

- la première concerne la ligne où sont présentées les commandes 9 et 10; cette ligne n'est pas homogène avec les autres puisqu'il y a deux commandes sur la même ligne ce qui la rend quasiment illisible.

- la seconde concerne l'écriture du chiffre zéro qui est présenté avec une barre, ce qui est usuel uniquement pour les informaticiens et inconnu pour tous les autres utilisateurs.

La figure 2.3 représente un des feuillets de l'aide à l'utilisation ; le premier feuillet de cette aide est obtenu en tapant 12 ou AI suivi de

"ENVOI" à la suite de l'écran SOMMAIRE; pour obtenir le deuxième feuillet, il faut appuyer sur 1 suivi de la touche "ENVOI" alors que pour obtenir les autres feuillets, il faut appuyer sur "SUITE" (hétérogénéité de la syntaxe).

La première remarque que l'on peut faire sur cet écran concerne le vocabulaire; certains mots de ce vocabulaire ont une signification pour l'informaticien mais pas pour les autres utilisateurs. Il s'agit de "fonctions", "accès", "page adjacente" (la faute d'orthographe sur adjacente est d'origine!).

La deuxième remarque a trait à l'ambiguïté du message décrivant "Retour au feuillet précédent"; ce message exprime en fait deux idées différentes, la première a trait effectivement au retour à la page précédente, la seconde a trait à la "page rubrique précédente", ce qui n'est pas très explicite.

La troisième remarque n'est pas directement visible sur l'écran puisqu'elle concerne la vérification de la cohérence entre les explications contenues dans l'aide et le logiciel lui-même. Cela concerne les touches SUITE et RETOUR dont l'emploi au cours de l'utilisation du logiciel n'est pas toujours possible et dont les effets ne sont pas toujours ceux attendus.

RTR		AIDE A L'UTILISATION	
FONCTIONS DE CONSULTATION		TOUCHE	
SUITE OU FEUILLET SUIVANT	passage au message suivant ou à la page suivante	SUITE	
RETOUR AU FEUILLET PRECEDENT	retour au message ou à la page précédente, associé à *, retour à la page rubrique précédente	RETOUR	
REPETITION DU FEUILLET	répétition de l'écran en cas de mauvaise transmission associé à un choix numérique accès à une page adjacente sans avoir à revenir au sommaire correspondant	REPET.	
		appuyez sur	SUITE
RTR	AI		

Fig 2.3

La figure 2.4 représente le premier écran du service "horaires" que l'on obtient en tapant la commande "HORA" comme il est indiqué dans la première ligne.

La première remarque concerne le vocabulaire des messages de service:

- le mot "relation" qui désigne une liaison ferroviaire n'est pas en usage dans le vocabulaire courant;

La deuxième remarque concerne la logique d'utilisation au niveau de cet écran.

On trouve en effet, regroupés dans le service "horaires", deux autres services, "les réductions" et "les dernières nouvelles", qu'il n'est pas du tout évident de placer à cet endroit.

En ce qui concerne "les réductions", l'utilisateur doit pouvoir y accéder directement sans passer par une arborescence; c'est donc un service qui doit être présenté au niveau du sommaire général. Si l'accès direct n'est pas possible, il faut le regrouper dans un service du type "informations générales" où l'utilisateur peut s'attendre à trouver des informations variées à caractère général et non pas dans un service spécifique comme "horaires"

RTR	HORAIRES	HORA
<p>HORAIRES "VILLE A VILLE" :</p> <p>-de toutes les relations desservies par trains ou autocars directs (Banlieue de Paris exclue) soit 160.000 relations,</p> <p>-des principales relations desservies par correspondance, soit 2.000 relations</p> <p>pour une date comprise dans les deux mois à venir . tapez 1</p>		
		ENVOI
" Si vous hésitez		GUIDE
" Les réductionsREDU		ENVOI
" Les dernières nouvelles DERN		ENVOI
(C) RTR 1987		

Fig 2.4

La figure 2.5 représente un écran de consultation d'horaires où l'utilisateur a essayé de revenir à la page précédente (liste des horaires) au moyen de la touche "RETOUR"; on voit alors que s'affiche sur la première ligne le message d'erreur "touche retour ineffective".

Ce message est en contradiction avec les explications contenues dans "l'aide à l'utilisation" où l'on indique que la touche "RETOUR" permet de revenir à la page précédente.

Si la règle énoncée ne peut, pour des raisons techniques, être respectée, il faut modifier le texte de l'aide et indiquer pour chaque écran les touches actives à ce niveau (notion de menu dynamique).

Mais avant de faire ces modifications, il y aurait lieu de vérifier s'il y a une véritable impossibilité technique à respecter cette règle qui a l'avantage de la simplicité et qui correspond à une touche fonction.

The screenshot shows a terminal window with a title bar that reads "TOUCHE RETOUR INEFFECTIVE". The window is divided into two sections. The top section has a header with "RTR" on the left and "HORAIRES" on the right. Below this, the text reads: "LOCALITES :", "de départ : TOULOUSE (31)", "d'arrivée : PARIS (75)", and "DATE DE DEPART : 221Ø". A horizontal line separates this from the bottom section, which contains the text: "Horaires sur la même relation :", "1- même sens à une autre date", "2- retour à la même date", "3- retour à une autre date", and "tapez le N° . +". In the bottom right corner, there is a button labeled "ENVOI".

Fig 2.5

La figure 2.6 correspond à l'écran de réservation obtenu directement à partir de l'écran "Sommaire".

On peut faire des remarques de surface au sujet de la syntaxe des phrases suivantes:

- "Vous désirez des : ASSISES"; alors qu'il y a suffisamment d'espace pour rajouter le mot "PLACES".

- "Vous pouvez : - autre résevation ou horaire".

Ce type de syntaxe inutilement incomplète constitue pour les utilisateurs une gêne importante qui affecte leur compréhension.

Mais la remarque essentielle sur cet écran concerne le traitement des erreurs. Nous sommes dans un cas d'erreur non détectée avec message d'erreur erroné.

En effet, quand l'utilisateur partant de TOULOUSE tape PARIS comme destination, l'ordinateur lui demande de choisir parmi une des gares parisiennes; cette question est parfaitement inutile puisqu'il n'a pas le choix effectif de cette gare! Cette information devrait donc être déduite automatiquement.

Si l'utilisateur se trompe sur la gare d'arrivée (ici, en tapant PARIS-NORD), le logiciel ne lui signale pas l'erreur et, en conséquence, l'utilisateur continue à saisir les autres données; après validation, il obtient comme message qu'il n'y a pas de train à l'heure indiquée.

Le comportement du programme est par ailleurs non stable car au cours d'un autre essai de ce type nous n'avons pas obtenu de message d'erreur mais un message d'attente "votre demande est transmise veuillez patienter" suivi d'une interruption pour dépassement de temps sans utilisation!

Quelle que soit la réponse du logiciel, l'utilisateur ne peut pas savoir qu'il a fait une erreur sur la gare de destination.

RTR **RESERVATION**

Vous voulez voyager :

- de : TOULOUSE-MATABIAU
- à : PARIS-NORD
- le : 241Ø
- par le train : ORDINAIRE DE Ø8HØØ

Vous désirez des : ASSISES

- en : 1. °classe
- nombre et préférences : 1 .

Nous ne pouvons effectuer votre réservation pour le motif suivant :
AUCUN TRAIN NE DESSERT LA RELATION
A L'HEURE INDIQUEE .

Vous pouvez :

- modifier votre demande1
- autre réservation ou horaire

ENVOI
SUITE

Fig 2.6

La figure 2.7 est un écran de réservation obtenu à partir du service "horaires", ce qui a pour conséquence que les quatre premières lignes sont remplies automatiquement par le logiciel car elles correspondent au choix fait par l'utilisateur sur l'écran précédent. Mais la zone indiquant le numéro du train n'est pas pertinente pour l'utilisateur alors que l'heure du train le serait et ne figure pas.

L'indication concernant la longueur du nom (limitée à 13 caractères) est une indication d'ordre technique qui est peu pertinente pour l'utilisateur; elle correspond à la logique de fonctionnement (indexation des fichiers) et non à la logique d'utilisation.

The screenshot shows a terminal-style interface for a reservation system. At the top, the title 'RTR' is on the left and 'RESERVATION' is on the right. The main content area contains the following text:

```
Vous voulez voyager :  
- de : TOULOUSE-MATABIAU  
- à : PARIS AUSTERLITZ  
- le : 241Ø  
- par le train : N 74  
Vous désirez des : ASSISES  
- en : 1. °classe  
- nombre et préférences : 1
```

Veillez nous indiquer en 13 caractères maximum votre nom qui permettra d e vous identifier, lors du retrait de votre titre de réservation :

Below the input field, there are three options with corresponding buttons:

après votre réponse	ENVOI
modifier la demande Ø	ENVOI
vous voulez être aidé	GUIDE

Fig. 2.7

6.3.2 Enchaînements d'écrans

Les enchaînements d'écrans peuvent être testés uniquement à partir du logiciel, mais l'analyse sera plus rapide si l'on dispose du dossier d'analyse et que l'on procède ensuite à des vérifications systématiques sur le logiciel. Sinon la première partie du diagnostic consiste à reconstituer l'arborescence du menu qui figure dans le dossier d'analyse. C'est ce qui a été fait dans ce cas et nous présentons ci-dessous deux schémas, avec le formalisme des automates d'états finis, qui ont été remplis lors des tests sur le logiciel RTR:

- le premier représente une partie de l'arborescence générale dans le sens descendant, c'est-à-dire en partant du menu général et en allant vers les écrans suivants; pour faire le lien avec les images d'écrans présentées précédemment, nous avons indiqué dans les places du schéma (représentées par un cercle) le numéro des figures correspondantes.

- le deuxième schéma représente les tests systématiques des touches RETOUR et SOMMAIRE pour chacun des écrans représentés sur le premier schéma; on voit cependant la touche GUIDE apparaître une fois quand on est dans le cas où les touches RETOUR et SOMMAIRE sont toutes les deux interdites; mais pour des raisons de lisibilité, il n'était pas possible de représenter sur le même schéma les tests systématiques de la touche GUIDE.

A partir de ces schémas, nous pouvons constater les dysfonctionnements suivants:

1. l'arborescence du menu ne correspond pas globalement à une logique d'utilisation; on peut à titre d'exemple citer les faits suivants:

- on peut aller de l'écran "horaires" vers l'écran "réservation" (comme on peut le voir sur le schéma représentant un extrait de l'arborescence de RTR) mais l'inverse n'est pas vrai; ce fait constitue une gêne importante car il est usuel d'avoir besoin de consulter des horaires en cours de réservation soit pour s'assurer de l'horaire précis soit pour redresser une erreur.

- le sommaire général (figure 2.2) s'adresse à des types d'utilisateurs différents et à des niveaux différents; il y aurait lieu de distinguer au minimum les voyageurs (horaires et réservations) et les entreprises (transports par wagons complets et en conteneurs). Les différents types d'information à caractère général peuvent être regroupés par type d'utilisateur (informations générales, actualité,...).

- certaines informations de niveau un ne sont pas accessibles directement; par exemple, les réductions ne sont accessibles qu'à partir des horaires, ce que l'utilisateur ne peut absolument pas deviner.

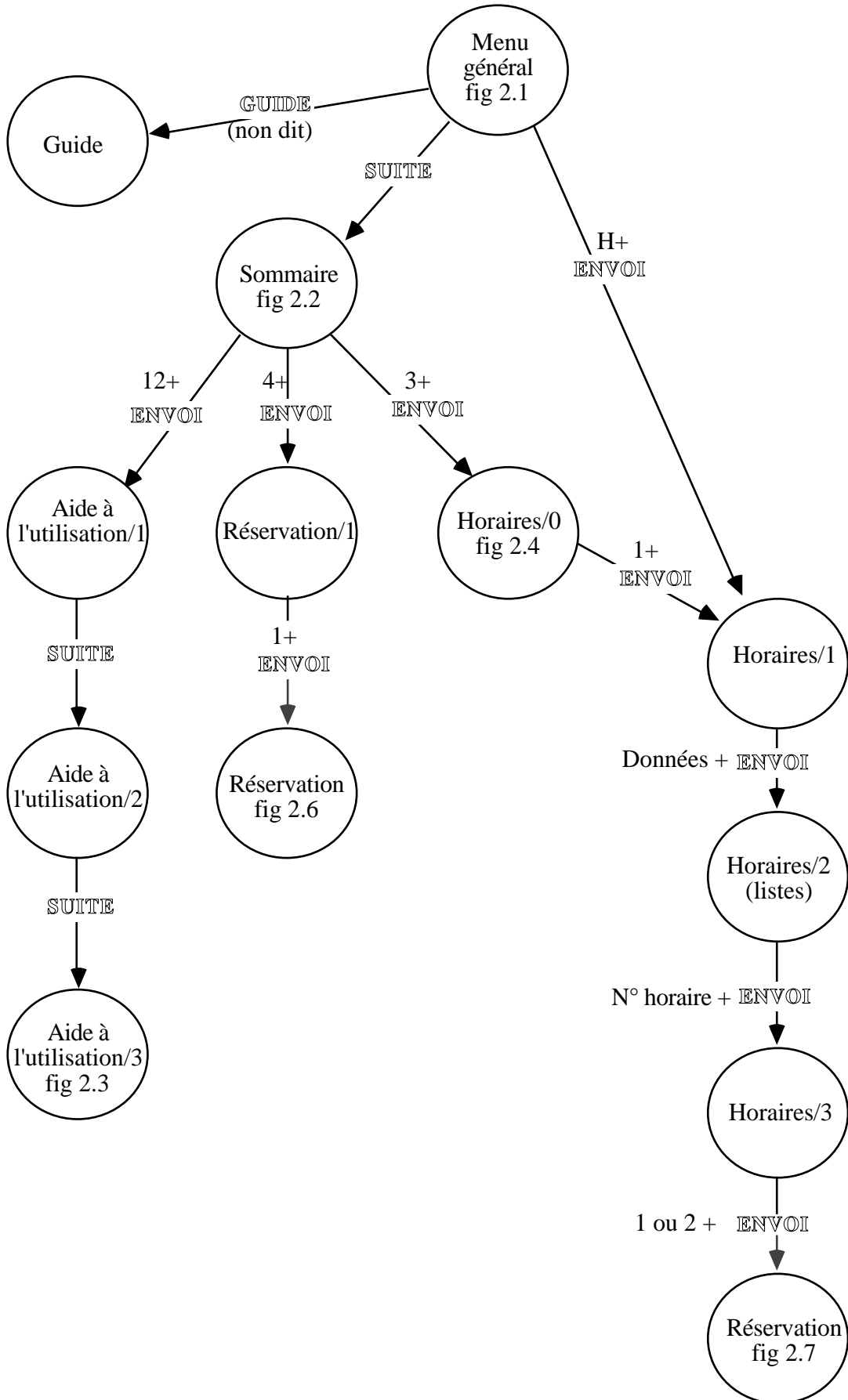
2. Les commandes standard d'enchaînement ne sont pas systématiquement activables par l'utilisateur, sans qu'aucune indication ne lui soit fournie à ce sujet. Il s'agit des commandes correspondants à des touches-fonctions comme SOMMAIRE, GUIDE, ANNULATION, CORRECTION, RETOUR, SUITE.

De plus, l'effet de ces commandes n'est pas toujours identique, ce qui rend le comportement du logiciel non prévisible pour l'utilisateur et lui interdit de développer un automatisme.

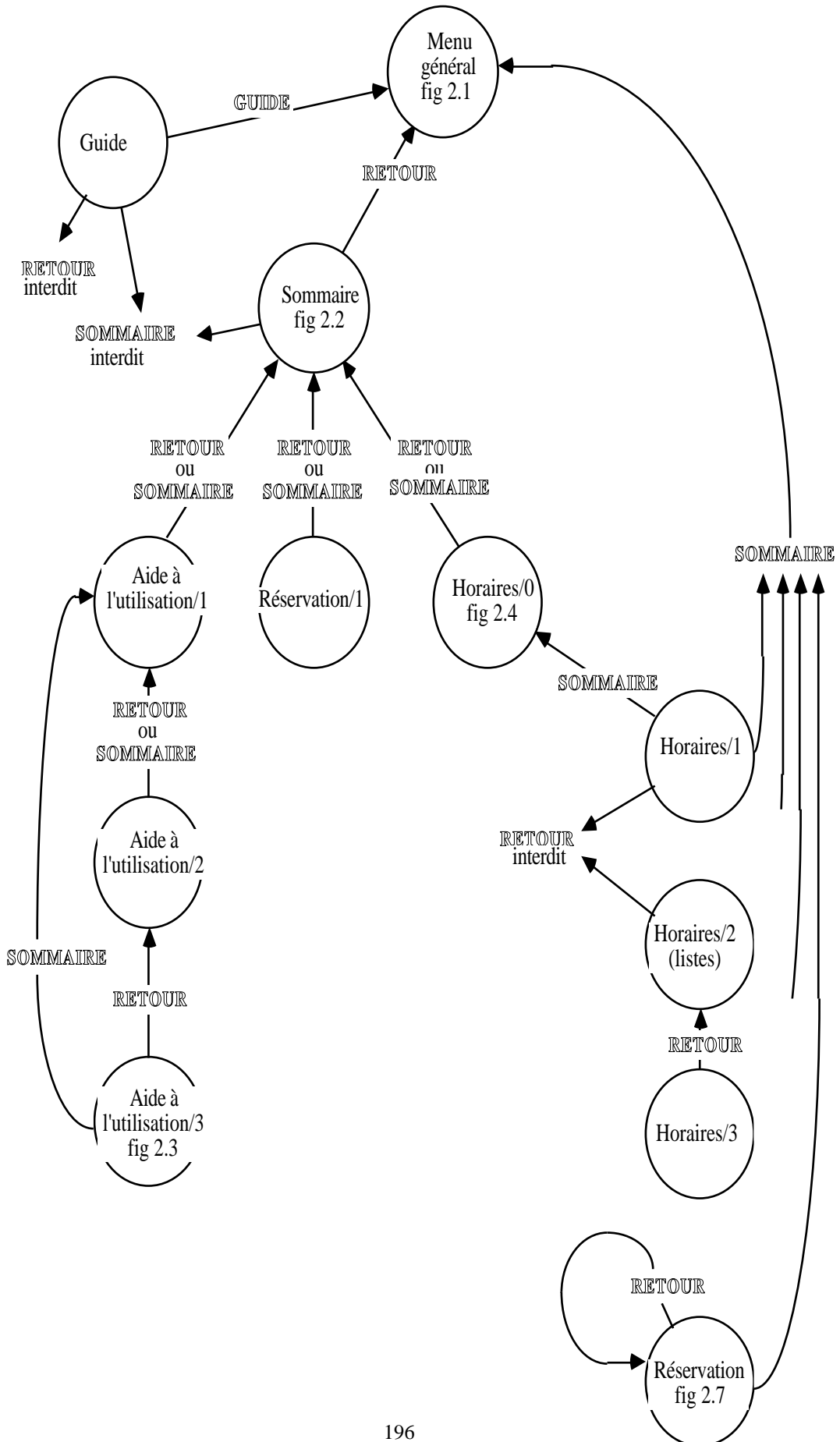
Nous pouvons voir sur le deuxième schéma que dans la branche "horaires", sur quatre écrans successifs, on a trois actions différentes pour la même touche RETOUR (interdiction, retour à l'écran précédent, retour à la ligne précédente du même écran). De même pour la touche SOMMAIRE qui fait parfois revenir à l'écran précédent, parfois deux écrans précédents, parfois au début de l'arborescence ou qui est interdite.

Comme nous l'avons indiqué au début de ce chapitre, le formalisme des automates d'états finis, qui a l'avantage d'être très facile à utiliser pour décrire les enchaînements d'écrans, présente une ambiguïté quand il s'agit d'exprimer le parallélisme et les synchronisations. Nous pouvons constater ce problème sur le deuxième schéma où, à partir de l'écran "Horaires/1", il y a deux sorties par la touche SOMMAIRE selon que l'on vient de "Horaires/0" ou de "Menu général". Pour lever cette ambiguïté due au formalisme, il faudrait dupliquer complètement la branche constituée des écrans "Horaires/1", "Horaires/2", "Horaires/3" et "Réservation"...; ceci alourdirait la présentation du schéma sans offrir d'avantages notables pour satisfaire l'objectif que l'on s'est fixé ici: le diagnostic du logiciel vu du point de vue de l'utilisateur.

Extrait de l'arborescence du menu



Test des touches RETOUR et SOMMAIRE



BIBLIOGRAPHIE

1 - Informatique

Livres

- (Coh, 82) COHEN et FEIZENBAUM
The Handbook of Artificial Intelligence - Vol. 3
Ed. PITMAN - 1982
- (Col, 86) A.COLLONGUES, J.HUGUES, B.LAROCHE
MERISE, méthode de conception
Dunod Informatique - 1986
- (Pol, 86) P.PELLAUMAIL
La méthode AXIAL, conception d'un système
d'information
Editions d'Organisation - 1986
- (Tar, 83) H. TARDIEU, S. ROCHFELD, R.COLETTI
La méthode MERISE
Ed. Organisations - 1983

Revues

Maquettage et prototypage : outils et techniques
Génie Logiciel n°3 - janvier 1986

Informatique de gestion : les méthodes
Génie Logiciel n°4 - avril 1986

Informatique de gestion : les outils
Génie Logiciel n°5 - juillet 1986

Articles

- (Bar, 86) MF. BARTHET, C. SIBERTIN-BLANC
La modélisation d'applications interactives adaptées aux
utilisateurs par des Réseaux de Petri à Structure de Données
Congrès AFCET, Génie Logiciel, Versailles, mai 86

- (Bra, 86) G. BRAJNIK, G. GUIDA, C. TASSO
An expert interface for effective man-machine interaction
Cooperative Interfaces to Information System,
SPRINGER-VERLAG - 1986
- (Cou, 86) J. COUTAZ
La construction d'interfaces homme-machine
Rapport IMAG n°635 - 1986
- (Mer, 81) MEYROWITZ et MOSER
Bruwin : an adaptable design strategy for window
manager/virtual terminal system.
ACM SIGOPS - Vol 15 N° 5 - December 1981
- (Sar, 74) SACERDOTI
Planning in a hierarchy of abstraction spaces.
Artificial Intelligence 5 - 1974
- (Sib, 85) SIBERTIN-BLANC
Les réseaux de Pétri de haut niveau comme formalisme
de modélisation des traitements d'un système
d'informatique. Convention Informatique Latine - 1985
- (Was, 82) WASSERMAN
The user software engineering methodology:
an overview
University of California, San Francisco - 1982
- (Weg, 83) WEGMANN
Vitrail - Conception et réalisation d'un noyau de
communication homme-machine pour un système
bureautique intégré.
Thèse Docteur Ingénieur - Université Paris VI -
Septembre 1983

2 - Ergonomie

Livres

- (Art, 81) ACTIF
Informatisation et vie au travail
Les Editions d'Organisation - 1981

- (Cac, 80) CACKIR, HART, STEWART
Les terminaux à écran
Editions d'Organisation - 1980
- (Not, 80) L.NORMAN
Traitement de l'information et comportement humain
Etudes Vivantes, Montreal - 1980
- (Spe, 80) SPERANDIO
La psychologie en ergonomie.
PUF - 1980

Articles

- (Bar, 87) MF. BARTHET, L.PINSKY
Analyse du travail ergonomique et méthodes d'analyse informatique
Les Cahiers "Technologie, Emploi, Travail", n°4
L'ergonomie des logiciels, Documentation Française -
octobre 87
- (Bis, 79) BISSERET
Utilisation de la théorie de la détection du signal pour
l'étude des décisions : effet de l'expérience
des opérateurs.
Rapport INRIA C. 07911R60 - 1979
- (Bou, 79) BOUJU - SPERANDIO
Analyse de l'activité visuelle des contrôleurs d'approche.
Rapport INRIA C. 07911R59 - 1979
- (Caz, 81) CAZAMIAN
Image et Action - Doc. interne - 1981
Département d'Ergonomie et d'Ecologie Humaine.
Université Paris I.
- (Cor, 82) CORSON
Aspects psychologiques liés à l'interrogation d'une
base de données.
Rapport INRIA N° 126 - Avril 1982
- (Der, 78) DERMOTT
Planning and Acting.
Cognitive Science - Vol. 2 - 1978
- (Gra, 81) GRAESSER, ROUANNET, DE LLENBACH
Incorporating Inferences in Narrative Representations :
A study of How and Why - Cognitive Psychology, 1981

- (Lep, 83) J.LEPLAT, JM. HOC
Tâche et activité dans l'analyse psychologique
des situations
Cahiers de psychologie cognitive - mars 1983
- (Och, 72) OCHANINE - QUAAS - ZALTMAN
Déformation fonctionnelle des images opératives.
Questions de psychologie N° 13 - 1972
- (Pin, 87) L.PINSKY
De l'interface à la situation de travail
Rapport introductif, congrès SELF, Liège -
septembre 1987
- (Ric, 83) JF. RICHARD
Logique du fonctionnement et logique d'utilisation
Rapport INRIA n°202 - avril 1983
- (Rol, 82) ROLLOY
Etude des conditions de travail en agence commerciale
des télécommunications.
Rapport interne - 1982
- (Sca, 87) D. SCAPIN
Guide ergonomique de conception des interfaces homme-
machine
Rapport INRIA n°77 - 1987
- (Seb, 87) S.SEBILLOTTE
La planification hiérarchique comme méthode d'analyse
de la tâche
Rapport INRIA n°599 - 1987
- (Sec, 87) B. SENACH
Intelligence des logiciels d'aide à l'utilisation
et modélisation de l'activité des utilisateurs
Les Cahiers "Technologie, Emploi, Travail", n°4
L'ergonomie des logiciels, Documentation Française -
octobre 1987

3 - Livres divers

- (Bra, 83) G.W.BRAMS
Réseaux de Petri
Masson - 1983

(Me1, 77) MELESE
Analyse modulaire des Systèmes
Ed. Hommes et Techniques - 1977

GLOSSAIRE

Activable

Propriété d'une opération ; une opération est activable si les conditions du pré-requis sont réalisées .

Activité

L'activité est ce qui est mis en oeuvre pour exécuter la tâche . C'est la trace de ce qui est fait réellement par l'utilisateur .

But

Etat final que doit atteindre le système homme-machine . Il sera décrit par des critères et la valeur qu'ils doivent prendre .

Déclenchement

Propriété d'une opération ; c'est un évènement particulier en entrée d'une opération ; si le déclenchement est optionnel, cela signifie que cet évènement est déclenché par l'utilisateur . Si le déclenchement est *systematique*, cela signifie que cet évènement est déclenché par l'ordinateur .

Entité

C'est la représentation abstraite d'un objet identifiable du monde réel . Une entité sera décrite par une liste de rubriques .

Evènement

C'est un fait dont l'apparition est de nature à déclencher l'exécution de traitements. Dans le cas de la description d'un système d'information, l'évènement sera décrit par un ensemble de données .

Facultative

Propriété d'une opération . Une opération est facultative si son exécution ou sa non-exécution n'interdit pas d'atteindre le but .

Indicative

Propriété d'une précédence . Une précédence est indicative si elle existe dans une ou plusieurs procédures effectives ou prescrites sans exister dans la procédure minimale .

Latitude décisionnelle

C'est la partie du pilotage de la tâche qui est sous la responsabilité de l'utilisateur .

Logique de fonctionnement

C'est un découpage de l'application du point de vue de l'informatique (analyste ou programmeur).

Logique d'utilisation

C'est un découpage de l'application du point de vue de l'utilisateur.

Obligatoire

Propriété d'une opération . Une opération est obligatoire si son exécution, quand elle est activable, est nécessaire pour atteindre le but .

Opération

C'est un ensemble de transactions qui peuvent être exécutées consécutivement et sans attente d'évènements externes au système homme-ordinateur .

Opération élémentaire

C'est la plus petite décomposition possible qui ait un sens pour l'opérateur.

Optionnel

Voir "déclenchement".

Permanente

Propriété d'une précedence . Une précedence est permanente si elle existe dans toutes les descriptions procédurales de la tâche .

Pilotage

Le pilotage d'une tâche, au sens de la théorie des systèmes, recouvre la notion de contrôle de l'exécution et de régulation de la tâche .

Poste de travail

C'est un ensemble de moyens mis à la disposition d'un utilisateur pour lui permettre de réaliser un ensemble de fonctions déterminées dans une organisation donnée .

Précédence

C'est un lien de priorité d'exécution entre opérations . Si $P(O_i, O_j)$ est satisfait, cela signifie que O_i doit être exécuté avant O_j .

Une précedence est *permanente* si son exécution est contrôlée par l'ordinateur; elle est *indicative* si son exécution est contrôlée par l'utilisateur.

Pré-requis

C'est l'ensemble des conditions (sur les évènements et sur les données) qui doivent être vérifiées pour que l'opération puisse s'exécuter .

Procédure

Description formelle et détaillée d'une tâche permettant de décrire l'enchaînement des différentes opérations .

Procédure effective (PE)

C'est une description permise d'une tâche ; c'est un modèle de l'activité .

Procédure minimale (PM)

C'est l'ensemble des opérations et des enchaînements minimaux pour que le but de la tâche puisse être considéré comme atteint par l'ordinateur .

Procédure prévue ou prescrite (PP)

C'est la description d'une tâche telle qu'elle est prévue dans son déroulement standard.

Règle d'émission

Les règles d'émission sont des règles incluses dans les opérations et dont les valeurs déterminent les événements de sortie de l'opération.

Représentation conceptuelle (RC)

C'est la représentation abstraite d'un poste de travail vu par l'analyste .

Représentation externe (RE)

C'est la représentation concrète des applications interactives vues par l'utilisateur .

Représentation interne (RI)

C'est la représentation concrète des applications interactives telles qu'elles seront programmées .

Rubrique

C'est la plus petite information significative qui puisse être utilisée de façon autonome.

Synchronisation

C'est une proposition logique booléenne portant sur la liste des événements d'entrées d'une opération .

Systematique

Voir "déclenchement"

Tâche

C'est une fonction d'un poste de travail . Elle sera décrite par un but et des conditions; les conditions peuvent être décrites soit par l'ensemble des états à parcourir pour atteindre le but, soit par les opérations admissibles pour parcourir ces états, soit par la procédure (ou combinaison de ces opérations) .

Transaction

C'est une interaction complète (un aller-retour) entre l'utilisateur et le système technique .