Un pas vers l'évaluation automatique des Interfaces Homme-Machine

Sandrine Balbo, Joëlle Coutaz

Laboratoire de Génie Informatique, IMAG
B.P. 53X, 38041 Grenoble Cedex
e-mail : balbo@imag.fr, joelle@imag.fr

Résumé :
L'évaluation de l'ergonomie d'un logiciel peut s'effectuer à l'étape de conception au moyen de modèles prédictifs. Elle peut également être appliquée au prototype exécutable selon diverses techniques d'observation. Dans ce cas, les données recueillies doivent être analysées. Notre objectif est de simplifier cette tâche en offrant une solution informatique. L'article décrit les points essentiels d'une expérimentation préliminaire : un modèle de tâche orienté flot de données, capture des actions de l'utilisateur, identification de règles de comportement et liaison avec un critique ergonomique.

Mots clés :
Capture du comportement, critique ergonomique, modèle de tâche, évaluation d'interface, Interface Homme-Machine.

Summary :
The evaluation of the usability and the learnability of a computer system may be performed with predictive models during the design phase. It may be done on the executable code as well by observing the end user in action. In this case, data collected in vivo must be processed. Our goal is to provide a software support for performing this difficult task. This article describes an early experience in this direction. It includes the description of a data-flow oriented task model, the acquisition of end user actions, the identification of heuristic rules that model patterns of behaviour, and their link to a software usability critic.

Key words :
Behaviour observation, critic, task model, evaluation, man-machine interface.

1. Introduction

L'ingénierie des Interfaces Homme-Machine (IHM) s'est appliquée pour l'essentiel, au développement d'outils d'aide à la construction de maquettes et de prototypes exécutables. En l'absence de techniques de conception plus robustes, la conception itérative est un recours naturel. Le cycle "conception-implémentation-évaluation" appliqué au prototype exécutable offre un terrain d'observation plus réaliste que, par exemple, les simulations papier. Toutefois, l'évaluation, lorsqu'elle s'effectue par dépouillement manuel des données observées, est une tâche longue, difficile, requérant expertise et patience. Notre objectif est de simplifier ce travail en offrant une solution informatique à l'étape d'analyse.

Cet article est organisé comme suit : la section 2 résume l'état de l'art dans le domaine de l'évaluation des interfaces. La section 3 introduit la plate-forme d'expérimentation. Les sections suivantes reprennent les points essentiels de notre étude : modèle de tâche, capture des actions, règles de comportement et critique ergonomique.

2. Etat de l'art

L'évaluation des interfaces homme-machine se pratique selon deux stratégies complémentaires : la simulation et l'observation. La première, prédictive, a lieu avant la mise en œuvre du prototype ; la seconde permet un constat sur le prototype lui-même.

En l'absence de théorie cognitive unificatrice, les systèmes d'évaluation prédictifs sont contraints d'adopter "un profil bas" : la portée des services offerts est faible ou bien la modélisation s'appuie sur des hypothèses réductrices. Comme exemple de service ciblé, citons KRI (Knowledge-based Review of user Interface) [Löwgren 90]. Ce système effectue l'analyse d'une description formelle de l'interface. Pour cela, il utilise une base de connaissances expertes ainsi que des règles exprimant des recommandations générales d'ergonomie (de type Smith&Mosier). KRI présente ses critiques sous forme de commentaires et propose des améliorations. Toutes les indications fournies relèvent de l'aspect syntaxique de l'interaction. La modélisation de la tâche, et plus généralement toute activité cognitive d'ordre supérieur, sont ignorées.

A l'inverse, PUM (Programmable User Models) prédit les erreurs en s'appuyant sur une architecture cognitive programmable [Young 90]. Le concepteur d'interface spécifie au moyen d'un langage idoine ce que l'utilisateur doit connaître pour accomplir une classe de tâches représentatives. Cette description, qui inclut des connaissances sur le domaine et sur les caractéristiques de l'interface, est compilée sous forme de règles. Cet ensemble de règles représente l'aptitude supposée de l'utilisateur à exécuter la tâche avec l'interface spécifiée. L'architecture PUM fait de son mieux pour produire un plan de résolution d'une tâche donnée. Une planification infructueuse doit attirer l'attention du concepteur sur un défaut potentiel d'utilisabilité ou une difficulté d'apprentissage.

A la différence de PUM qui construit dynamiquement un plan de résolution, GOMS [Card 83] et ses dérivés telle la "Théorie de la Complexité Cognitive" [Kieras 85], s'appuient sur une décomposition explicite des tâches en sous-tâches. Cette décomposition, statique, est supposée représenter l'approche de résolution de l'utilisateur. Le GOMS original est un pur modèle prédictif des performances qui ne tient pas compte des erreurs. Plusieurs études ont cherché à étendre GOMS à la prédiction des erreurs liées notamment à la surcharge cognitive [Lerch 89]. La Théorie de la Complexité, quant à elle, permet de comparer les mérites relatifs de plusieurs conceptions en termes de transfert de connaissances et de difficulté d'apprentissage.

Tous ces modèles sont prédictifs. Ils ont l'avantage de détecter des anomalies de conception avant la mise en œuvre qui, malgré l'avancée des outils de prototypage, reste coûteuse. Ils ont aussi leurs inconvénients : spécifier les données nécessaires au modèle peut prendre autant de temps que programmer l'interface lui-même. En outre, ces données s'appuient sur des supposés, non pas sur des certitudes.

L'observation, qui constitue la seconde stratégie d'évaluation, capture des données réelles centrées sur le comportement perceptible de l'utilisateur. Les hypothèses de l'approche prédictive sont alors remplacées par des faits. De ce point de vue, l'évaluation tend à gagner en précision mais elle intervient après l'étape de conception, une fois le prototype réalisé. L'évaluation fondée sur la capture du comportement perceptible de l'utilisateur a deux finalités possibles : l'adaptativité et la critique.

L'adaptativité est la capacité du système à s'adapter automatiquement aux conditions de l'interaction. Elle s'applique notamment à la migration de tâches comme dans Opal [Hardman 89], à la négociation comme dans Sphinx [Lee 90]. La migration concerne le glissement automatique des responsabilités entre le système et l'utilisateur pour la réalisation de certaines tâches. Ce problème se trouve souvent abordé dans les systèmes de contrôle de processus où les plans de conduite sont généralement bien cernés. Concernant les tâches de conception telle l'édition de document, quelques expérimentations simples ont été effectuées. Eager en est un exemple [Cypher 91]. Ce système, réalisé au-dessus d'HyperCard, détecte les tâches répétitives et, par suite, construit automatiquement les macrocommandes correspondant à ces tâches. La négociation est un métadialogue le plus souvent engagé par le système en vue de lever les ambiguïtés. Elle tend à rendre le dialogue plus robuste. Elle est souvent utilisée dans les systèmes de reconnaissance de la langue naturelle pour pallier les insuffisances technologiques.

Dans notre cas, la capture du comportement perceptible de l'utilisateur va nous servir à construire un critique. Nous avons présenté des exemples de critique prédictif tel KRI. Nous ne connaissons pas de critique fondé sur les observations réelles. Ce critique de l'interface utilisateur se définit selon quatre facettes : (a) le modèle de tâche, référence d'utilisation optimale de l'interface ; (b) l'observation dynamique des actions de l'utilisateur en vue de la mise en place d'un modèle comportemental [Balacheff 91] ; (c) l'identification de règles de comportement ; (d) l'interprétation ergonomique des observations.

Avant de reprendre chacun de ces points dans les sections qui suivent, nous devons présenter Compo, le système interactif qui nous sert de plate-forme d'expérimentation.

3. Compo : la plate-forme d'expérimentation

Compo est un environnement auteur pour la construction d'applications embarquées sur des véhicules de tourisme. Ce système doit être utilisable par des concepteurs d'interface non informaticiens (et notamment par des ergonomes). Il doit permettre la conception d'interfaces de type manipulation directe à plusieurs fils. A cette fin, la construction d'une application embarquée s'effectue, elle aussi, de manière interactive par manipulation directe et par démonstration : les éléments se dessinent et les animations se montrent (elles ne se spécifient pas dans un langage). Toutefois, la programmation devient nécessaire lorsque certaines entités de l'interface ont un comportement non prévu dans l'environnement de développement. Nous espérons faciliter cette tâche à des non informaticiens en proposant un langage original de nature iconique [Chabert 91].

L'application que le concepteur construit obéit à une certaine structure. En effet, une application embarquée est constituée d'un ensemble de scénarios, par exemple un scénario "aide à la navigation", un scénario "réglages", etc. A son tour, un scénario comprend des groupements d'objets, qui correspondent approximativement à la notion de page écran. Par exemple, le scénario "réglages" inclut un groupement d'objets pour régler le chauffage, un autre pour régler l'auto-radio, etc. Chaque objet d'un groupement représente un concept du domaine de la voiture.

L'IHM de Compo doit permettre au concepteur d'applications embarquées de manipuler (observer, dessiner, programmer) les notions d'application, de scénario, de groupements, etc. gérées par l'environnement. Pour un groupement d'objets, il est notamment possible d'ouvrir une vue structurelle, une vue logique, et une vue graphique. La vue structurelle d'un groupement montre sous forme d'icône les objets qui le constituent. Sa vue graphique est le script en langage iconique qui spécifie le comportement dynamique du groupement (par exemple les enchaînements entre les groupements ou le changement de couleur d'un objet). Sa vue graphique est la restitution du groupement tel qu'il apparaîtra sur l'écran du véhicule. Toutes ces vues sont éditables : l'une pour ajouter/retirer un objet dans le groupement, l'autre pour modifier le script, la dernière pour dessiner à façon les formes des objets et décider des couleurs.

Compo offre un terrain d'expérience pour l'observation de tâche de conception (celle d'applications embarquées). Une tâche de conception se caractérise par l'absence d'ordre strict et stable des sous-tâches qui la composent. Par opposition au contrôle de processus, une tâche de conception révèle un comportement opportuniste [Hayes-Roth 79] qui se manifeste par l'entrelacement de tâches et des retours arrière. Compo n'est donc pas un cas d'école réducteur.

Ce système auteur a été réalisé sur station Sun au-dessus de X Window/Motif.

4. Modèle de tâche : arbre des séquences

Un modèle ne se conçoit pas indépendamment de sa finalité. Ce point est abordé au paragraphe suivant avant de présenter notre choix : l'arbre des séquences.

4.1. Finalité d'un modèle de tâche

La finalité d'un modèle de tâche diffère selon la phase du cycle de vie d'un logiciel. Dans les phases préliminaires de conception, le modèle de tâche exprime la logique d'utilisation du futur système. Comme le préconisent la méthode Diane [Barthet 88] et bien d'autres, la modélisation s'appuie sur une analyse du travail centrée sur l'utilisateur [Norman 86]. Dans certains cas, comme dans Sirocco [Normand 92], le modèle de tâche peut donner lieu à la génération automatique d'une IHM. Dans le cas général, le modèle structure l'espace de travail en termes de tâches et sous-tâches montrant les relations entre des grappes d'activité logiquement connectées. On obtient ainsi une représentation arborescente dont les feuilles représentent les tâches élémentaires, c'est-à-dire les tâches conceptuellement insécables. Appelons cette représentation, le modèle de tâche conceptuel.

En phase de test du prototype, un modèle de tâche vise à représenter la logique de fonctionnement du système. Contrairement au modèle conceptuel, il ne définit pas la logique d'utilisation souhaitable pour l'utilisateur, mais la façon idéale du point de vue du concepteur de conduire le travail avec le système réel. Dans le cas idéal, le modèle de tâche conceptuel est isomorphe au modèle de tâche effectif du prototype. En particulier, une tâche élémentaire du modèle conceptuel se traduit par une commande dans le modèle de tâche effectif.

Si l'on perçoit assez bien le lien entre la notion de tâche élémentaire du modèle conceptuel et celle de commande dans le modèle de tâche effectif, la correspondance pour les tâches composites est au contraire difficile à formaliser. Il y a plusieurs raisons à cela : (a) Sauf exception, le passage du modèle conceptuel à la mise en œuvre du système est manuel. L'isomorphisme entre les deux modèles ne peut donc être prouvé ; (b) les tâches considérées en phase de conception ont trait au domaine. Elles n'incluent pas les tâches para-, telle la manipulation des fenêtres, induites par l'utilisation du support informatique.

Il est clair que dans notre cas nous devons considérer l'étude d'un modèle de tâche effectif : l'arbre des séquences.

4.2. L'arbre des séquences

En général, les modèles de tâches s'appuient, nous l'avons dit, sur une approche descendante de décomposition de tâches complexes en grappes de tâches plus simples. Le formalisme de représentation est choisi en conséquence. Par exemple, l'arbre lexico-temporel de MAD (Méthode Analytique de Description des tâches) [Scapin 89] représente les décompositions successives des tâches en sous-tâches de plus en plus fines jusqu'à atteindre les tâches élémentaires. Cette approche convient en phase de conception et se calque parfaitement sur une planification hiérarchique de résolution de problème. Dans un domaine où les tâches sont de type conception, le comportement opportuniste de l'utilisateur s'accomode mal d'une planification hiérarchique. On observe au contraire plusieurs fils d'activité menées de manière

entrelacée. En partant de la notion de fil d'activité ("dialogue thread"), nous avons abouti à une modélisation de type flot d'information : l'arbre des séquences.

Un arbre de séquences représente les suites de tâches élémentaires que l'utilisateur peut entreprendre. Une tâche élémentaire, appelée également commande, est insécable. Sa réalisation par l'utilisateur entraîne une modification de l'état du système. En ingénierie des IHM et en première approximation, l'état du système est réparti sur deux composants logiciels : le noyau fonctionnel et l'IHM. Le noyau fonctionnel implémente les concepts et fonctions du domaine. Dans ce noyau, les techniques de représentation et de manipulation des concepts sont sensées être libres des techniques de représentation de ces concepts dans le composant IHM. Un mécanisme de transformation et de correspondance entre les représentations internes du noyau fonctionnel et les représentations perceptibles de l'IHM est assuré quelque part dans l'architecture du système interactif [Coutaz 91].

Les tâches élémentaires liées au domaine entraînent un changement d'état du noyau fonctionnel. D'autres ont une incidence uniquement sur l'état du composant IHM. Moran les dénomme "tâches syntaxiques" [Moran 81]. Elles correspondent, dans la terminologie de Harrison, à la notion de "commande passive" [Harrison 87].

Figure 1 : Extrait de l'arbre des séquences de Compo décrivant l'accès à une vue graphique de groupement d'objets.

La figure 1 montre un extrait de l'arbre des séquences établi pour Compo. Ce sous-arbre décrit l'accès à la vue graphique d'un groupement d'objets. Le formalisme choisi est le suivant. Un nœud représente une tâche élémentaire. La notation **A** ⊢→ **B** signifie que l'exécution de la tâche élémentaire **B** ne pourra avoir lieu que si la tâche élémentaire **A** a déjà été exécutée et une fois **A** dépassé, seul le sous-arbre de racine **B** sera accessible. La notation **A** → **B** signifie que l'exécution de la tâche élémentaire **B** ne pourra avoir lieu que si la tâche élémentaire **A** a déjà été exécutée avec retour possible ultérieur sur **A**. Lorsqu'un nœud a deux fils, l'utilisateur peut

continuer dans l'une ou l'autre des branches. Il peut à tout moment décider de passer d'une tâche à une autre du moment que les tâches prérequises ont été réalisées et que la précondition associée à la tâche est vérifiée. Une précondition est dénotée par le symbole **P:**.

Du point de vue de l'observateur, que celui-ci soit humain ou logiciel, la réalisation d'une tâche élémentaire se traduit par une suite d'actions physiques perceptibles produites par l'utilisateur. La figure 2 montre des exemples de correspondance entre des tâches élémentaires et les actions physiques nécessaires à leur accomplissement.

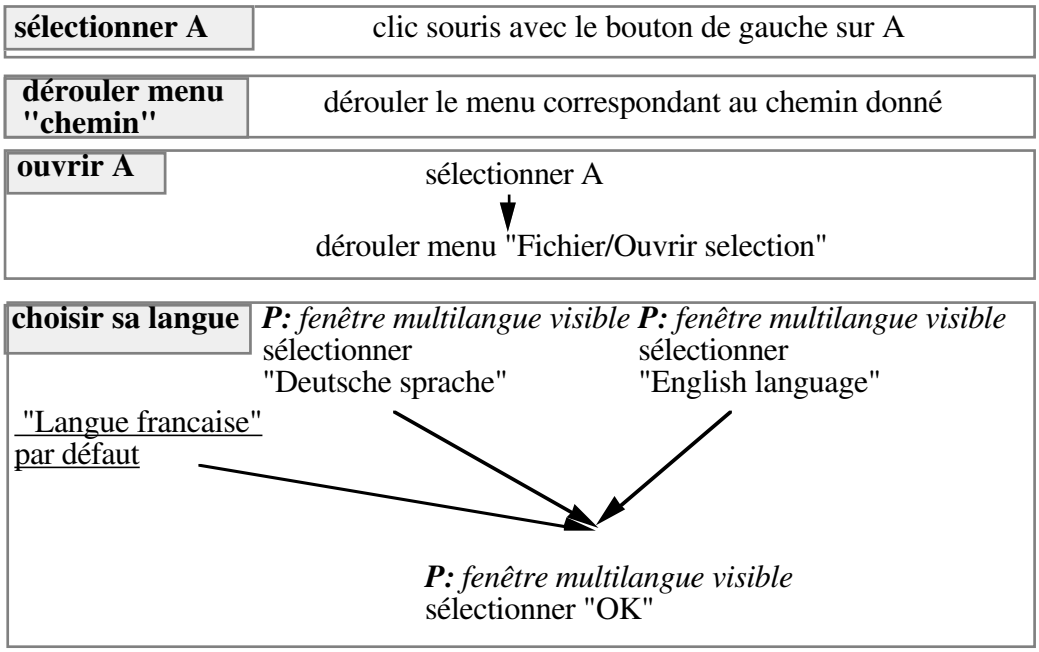

Figure 2 : Exemples de correspondance entre tâches élémentaire et tâches physiques.

La section suivante précise la nature des actions observées.

5. Capture des actions

Après avoir défini la notion d'action et son lien avec l'équivalent informatique (l'événement), nous indiquons comment nous la repérons dans l'univers X Window et comment nous la modélisons dans un fichier d'observation.

5.1. Notion d'action

L'action prend part au comportement. Selon J. Caelen et M. L. Fréchet, le comportement présente "une composante mentale qui se manifeste par des attitudes cognitives et une composante physique qui se manifeste par des actions" [Caelen 92]. Une action est donc un acte perceptible, conséquence des messages du processeur cognitif en direction du système moteur [Card 83].

Dans le cadre de notre étude, nous distinguons deux classes d'action : les actions observées et les actions ignorées. Les actions observées font l'objet d'une capture automatique par le logiciel tandis que les actions ignorées ne sont pas considérées par le système. Actuellement, les actions observées sont limitées à l'usage du clavier et de la souris. Progressivement et en collaboration avec d'autres équipes, nous

aborderons les formes langagières de la communication selon des modalités moins usuelles : parole, expression faciale et gestuelle.

L'évaluation d'une interface ne peut se satisfaire de l'observation des seules actions sur le clavier et la souris. Le méta-discours a aussi sa part dans la communication : commentaires oraux, mouvements du regard [Jacob 90], localisation de la bouche [Yuille 91] ou expression faciale [Terzopoulos 90]. Ces actes constituent une source importante d'information dont la fusion doit servir à lever des incertitudes sur l'interprétation des intentions de l'utilisateur. On constate une situation analogue en robotique mobile où les données en provenance de capteurs hétérogènes sont fusionnées en vue d'augmenter la robustesse du système.

Nous avons défini l'action comme la manifestation externe d'une décision cognitive. Puisqu'elle doit être observée par le système, nous devons lui trouver un modèle informatique : l'événement.

5.2. Notion d'événement

M.F. Barthet dans [Barthet 88] définit un événement comme "un fait dont l'apparition est de nature à déclencher l'exécution de traitement". Nous reprenons cette définition.

Les événements se répartissent en deux catégories selon les composants logiciels affectés : (a) les événements à destination du composant IHM. Leur traitement entraîne une modification de l'état du composant IHM et seulement du composant IHM. Ils modélisent les actions nécessaires à l'accomplissement de tâches syntaxiques. De ce fait, nous appelerons ces actions, des actions syntaxiques ; (b) les événements à destination du noyau fonctionnel entraînent un changement d'état du noyau fonctionnel accompagné éventuellement (et c'est souhaitable) d'un effet de bord sur l'état du composant IHM. Ces événements modélisent les actions nécessaires à l'accomplissement de tâches élémentaires spécifiques au domaine. De ce fait, nous appelerons ces actions, des actions sémantiques.

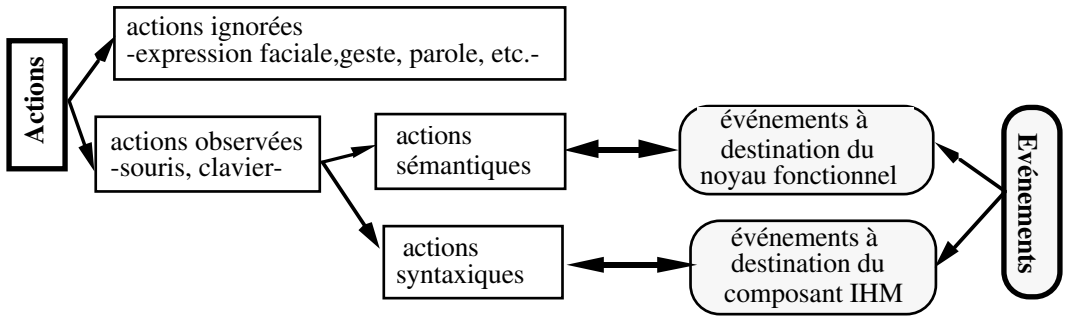

Figure 3 : L'espace des actions (rectangles à fond blanc), l'espace des événements (rectangles à fond grisé), et leur relation (double flèche). Les flèches simples expriment des relations entre classe et sous-classes.

La figure 3 résume les relations entre l'espace des actions et celui des événements. Le paragraphe suivant explicite leur lien avec l'environnement X Window.

5.3. X Window, Action et événement

X Window est un système de gestion des ressources physiques de l'interaction (clavier, souris, écran). Nous sommes concernés par trois de ses principes directeurs. (a) la distinction entre les fonctions de gestion (le serveur X) et la présentation des ressources (le "window manager") ; (b) la notion de procédure callback ; (c) les services d'ordre supérieur.

action	effet
sur la barre du haut, glissement avec le bouton gauche de la souris enfoncé	déplacement de la fenêtre
clic souris avec le bouton de gauche sur	iconification
clic souris avec le bouton de gauche sur	agrandissement plein écran de la fenêtre OU retour à une fenêtre de taille normale
clic souris avec le bouton de gauche dans la fenêtre	passage en avant-plan de la fenêtre

Figure 4 : Effet des actions sur les éléments décoratifs de mwm.

Concernant la séparation des rôles, le serveur X se charge par exemple de régler les recouvrements des fenêtres tandis que les éléments décoratifs de ces fenêtres (bouton de fermeture, bandeau de titre, etc.) et la façon de les manipuler sont fixés par le "window manager" préféré de l'utilisateur. Pour Compo, nous avons choisi le "window manager" *mwm* (Motif Window Manager).

La technique des procédures Callback est une façon de réaliser des systèmes réactifs. A un événement est associé un traitement. Ce traitement est encapsulé dans une procédure déclenchée automatiquement à l'occurrence de l'événement.

Par définition, les fonctions offertes par le serveur sont élémentaires. Aussi, des objets de présentation d'ordre supérieur, appelés widgets, (menus, barres de défilement, boutons, etc.), sont-ils fournis dans des boîtes à outils développées au-dessus du serveur. Pour Compo, nous avons choisi *Motif*. Un widget de classe *topLevel*, désigne une fenêtre reconnue par le window manager. Le contenu d'un topLevel peut être un ensemble de widgets mais ceux-ci ne sont pas reconnus par le window manager.

Cette dichotomie des fonctions entre serveur et window manager, de même la notion de toplevel et de widget, nous permettent de faire la distinction entre actions sémantiques et actions syntaxiques. Les actions syntaxiques sont identifiées à partir des événements captés par mwm et par certains widgets motifs comme les barres de défilement. Le tableau de la figure 4 explicite l'effet de quelques une des actions possibles sur les éléments décoratifs que mwm ajoute à un widget topLevel. La figure 5 montre un exemple de topLevel décoré par mwm.

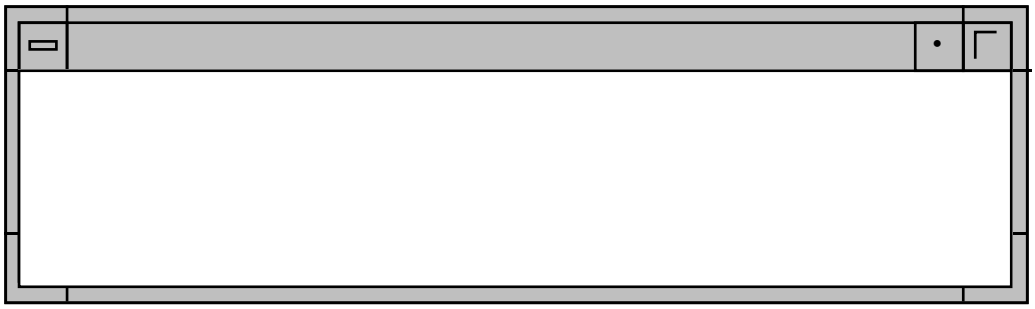

Figure 5 : Un topLevel et ses décorations mwm.

Les actions sémantiques correspondent aux événements dont les callbacks font référence à des fonctions du noyau fonctionnel.

Nous sommes maintenant en mesure de construire le fichier d'observation.

5.4. Fichier d'observation

Chaque action observée laisse une trace dans un fichier d'observations. Une trace est une information estampillée dont le format dépend de la classe de l'action. Une estampille est l'heure absolue d'occurrence de l'événement représentatif de l'action. Nous distinguons les actions sémantiques, les créations de fenêtres (les topLevel), les actions syntaxiques détectées par l'intermédiaire de mwm et celles relevées via certains widgets motif.

Le format du fichier d'observation est le suivant :

ligne 1 : **START THE APPLICATION AT :: *hh*:*mm*:*ss* :: *sec*+*s100***

Les autres lignes du fichier respectent l'un des formats suivants :

Action de nature sémantique :
***ss*+*s100*#*HH*:*MM*:*SS*:*S100*#**(*no*)**nom_cb**nom_de_l'action**

Création de topLevel :
OPEN TOPLEVEL : ***nom_fen*** **(*no*) ... DATE :** ***ss*+*s100***

Action de nature syntaxique gérée par mwm :
***ss*+*s100*##**(*no*)**type_de_l'événement_X**

Action de nature syntaxique gérée par motif :
***ss*+*s100*##**(*no*)**type_de_l'action_syntaxique**

où :

sec est le nombre de secondes écoulées depuis le 1er janvier 1970,
hh:mm:ss donne l'heure système,
ss et ***s100*** donnent le nombre de secondes et millisecondes écoulées entre le lancement du logiciel et l'enregistrement de l'action,
HH, ***MM***, ***SS*** et ***S100*** donnent la date, lorsqu'elle est connue, de l'événement associé à l'action (date donnée en millisecondes),
nom_fen donne le titre associé à la fenêtre créée,
no reprend le numéro associé au topLevel correspondant à la fenêtre,

type_de_l'événement_X est le type X de l'événement
nom_de_l'action reprend le nom d'une tâche élémentaire de l'arbre des séquences,
type_de_l'action_syntaxique donne le type de l'action syntaxique gérée par motif,
nom_cb est le nom de la callback appelée pour le traitement

La section suivante résume les retombées de l'analyse d'un tel fichier.

6. Règles de comportement

L'analyse manuelle de fichiers d'observation nous a permis d'identifier des "patterns" de comportement. A ces patterns nous avons fait correspondre des règles heuristiques utilisables pour le dépouillement automatique. Nous présentons ces règles dans le paragraphe ci-dessous suivi d'une illustration de cheminement dans l'arbre des séquences.

6.1. Un ensemble préliminaire de règles

Chacune des classes d'action envisagées se voit analysée par des règles spécifiques.

Règles portant sur les actions de nature sémantique :

Règle 1 : Changement de direction dans l'arbre des séquences.
Nous appelons changement de direction, les changements opérés par l'utilisateur lors de la "descente" dans l'arbre des séquences. Un changement de direction se traduit soit par un retour en arrière soit par un changement de branche dans l'arbre.

Règle 2 : Annulation immédiate d'une action.

Règle portant sur les actions de nature syntaxique :

Règle 3 : Répétition d'actions.
Exemples de répétition d'actions observées : modification de la taille ou iconification systématique de certaines fenêtres, manipulation fréquente des barres de défilement.

6.2. Un exemple de cheminement dans l'arbre des séquences

On appelle évolution, un cheminement temporel dans l'arbre des séquences. Une évolution est constituée d'une suite temporelle d'actions extraites du fichier d'observation. L'"évolution" de la figure 6 montre la suite d'actions effectuées par un utilisateur pour aboutir à l'ouverture de la vue graphique du groupement de nom "curseur1" appartenant au scénario "pap-c1" de l'application "demopape".

Ci-dessous une description textuelle de l'interaction correspondant à l'"évolution" de la figure 6 où figurent également les réactions pertinentes du système. La notation utilisée est la suivante :

n:: • (n° de fenêtre), événement : réaction associée avec entre parenthèses les numéros des nouvelles fenêtres créées

+ signifie l'apparition de la fenêtre dont le numéro est précisé entre parenthèses, - signifie la disparition de la fenêtre indiquée.

1:: • lancement du logiciel Compo : fenêtre choix de langue +(1)
2:: • (1), bouton radio"langue française" sélectionné : PARIS apparaît dans le cadre
3:: • (1), sélection du bouton "OK" : fenêtre -(1), Palette principale +(2)
4:: • (2), sélection du bouton radio "Application" : Palette des applications +(3)
5:: • (3), sélection du bouton "demopape" : bouton "demopape" sélectionné
6:: • (3), menu déroulant "Fichier/Ouvrir sélection" : APPLICATION:: demopape +(4)
7:: • (4), sélection du bouton "pap-c1" : bouton "pap-c1" sélectionné
8:: • (4), menu déroulant "Fichier/Ouvrir sélection" : SCENARIO::pap-c1 +(5)
9:: • (5), sélection du bouton "curseur1" : bouton "curseur1" sélectionné
10:: • (5), menu déroulant "Fichier/Ouvrir sélection" : GROUPEMENT::curseur1 +(6)
11:: • (6), sélection du bouton "Chauffage" : bouton "chauffage" sélectionné
12:: • (6), menu déroulant "Options/Vue Graphique" : Vue graphique de curseur1 +(7)

On remarque que la sélection du bouton "chauffage" à l'étape 11 est inutile par rapport à l'objectif "accès à la vue graphique de curseur1". En se reportant à l'arbre des séquences, l'action superflue va de pair avec un changement de direction et se manifeste comme un cul-de-sac. L'utilisateur ne donne pas suite à la sélection (voir Figure 6).

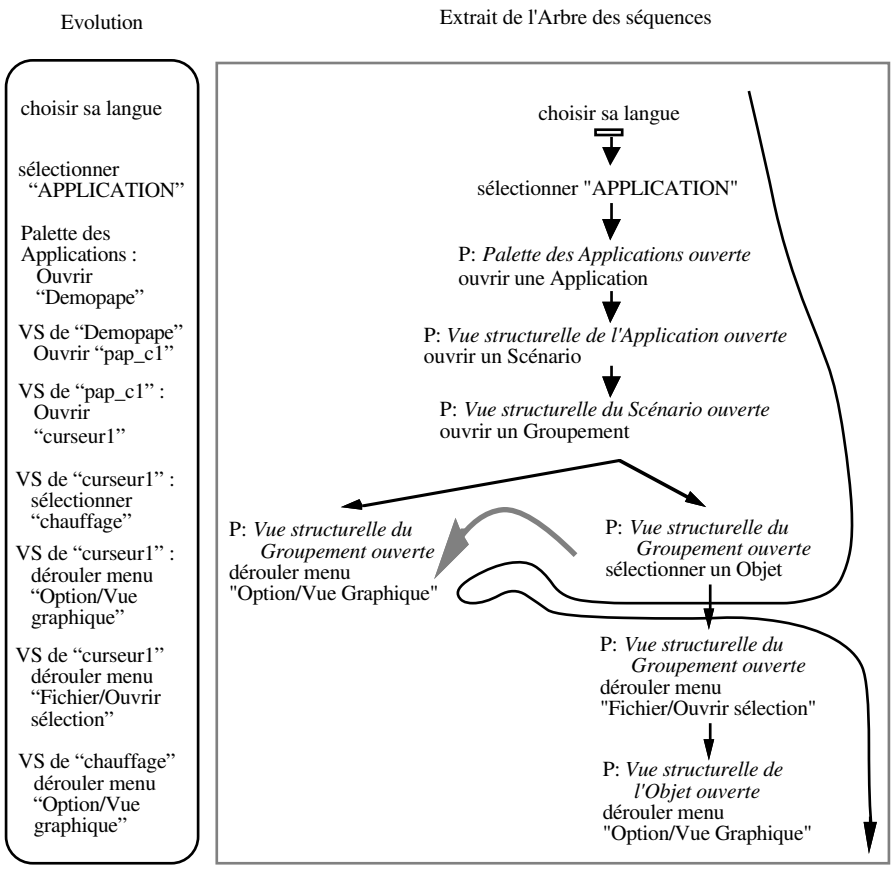

Figure 6 : Evolution. Dans ce schéma, la flèche noire représente la tâche effective. La flèche grise dénote un changement de direction observé.

Nous allons maintenant identifier le lien entre nos règles de comportement et les recommandations ergonomiques les plus courantes. Ce rapprochement est nécessaire à la construction du critique.

7. Critique : liens entre principes ergonomiques et règles de comportement

Les principes ergonomiques sont nombreux, voire inflationnistes et contradictoires. Nous visons dans cette première étape de réalisation du critique, à tester la faisabilité de l'entreprise. Aussi, allons-nous considérer un petit sous-ensemble de principes très généraux telles les recommandations de D.L. Scapin dans [Scapin 86]. On retrouve chez M.-F. Barthet des paramètres de conception qui recoupent les principes ergonomiques proposés par D.L. Scapin : la compatibilité, l'homogénéité, la concision, la flexibilité, le feed-back et le guidage, la charge informationnelle, le contrôle explicite ainsi que la gestion des erreurs.

La compatibilité suppose un faible recodage des informations entre le savoir de l'utilisateur et le format imposé par le logiciel. L'homogénéité s'apparente à la notion de cohérence (ex: séquence de commandes identiques pour un même résultat). La concision permet de réduire la charge mnésique (mémoire à court terme). La flexibilité propose différents niveaux d'utilisation du logiciel. Le feed-back et le guidage offrent une information rapide et pertinente sur l'état du système. La charge informationnelle doit être légère. L'utilisateur doit contrôler explicitement le logiciel et si tel n'est pas le cas, le logiciel doit en donner l'illusion. La gestion des erreurs doit réduire les occasions d'erreur et toute erreur doit être détectable dès son occurrence et doit pouvoir être corrigée.

Le tableau de la figure 7 traduit les relations possibles entre les règles de comportement et ces quelques principes ergonomiques. Voici l'interprétation que nous en faisons.

Principes ergonomiques / Règles	Compatibilité	Homogénéité	Concision	Flexibilité	Feed-back Guidage	Contrôle explicite	Charge informationnelle	Gestion des erreurs
Changement de direction	X	X			X		X	
Annulation immédiate		X	X	X			X	X
Répétition d'actions					X			

Figure 7 : Correspondance entre règles de comportement et principes ergonomiques. Une croix à l'intersection de la ligne i et de la colonne j signifie que la règle de comportement de la ligne i permet la

détection d'un problème relevant du principe ergonomique dela colonne j. La colonne du contrôle explicite est en grisé pour exprimer l'absence de règle étudiant ce cas.

Changement de direction et présence d'un cul de sac

La présence répétée d'un même cul-de-sac indique que l'interface conduit l'utilisateur à une situation d'impasse. Il y a incompatibité entre le modèle mental et le modèle de tâche du système. Dans le cas de Compo, nous avons constaté que l'incohérence de l'interface était la cause profonde de cette discordance.

Annulation immédiate d'une action

L'annulation immédiate d'une action traduit un problème de navigation.

Répétition d'actions syntaxiques

La répétition d'actions syntaxiques traduit une surcharge. En effet, les tâches syntaxiques, qui ne changent pas l'état du noyau fonctionnel, ne font pas évoluer l'objet de la tâche. On peut raisonnablement penser que la modification systématique de la taille des fenêtres dès leur ouverture traduit un mauvais choix sur leur taille initiale. On peut également inférer que les iconifications de fenêtres ou leur déplacement sont liés à l'encombrement de l'écran (problème du "messy screen").

Le tableau de correspondance donne un aperçu de la technique adoptée pour interpréter l'observation comportementale en termes de valeurs ergonomiques. Toutefois un critique logiciel aussi sophistiqué soit-il ne peut prétendre à la complétude. Il doit être vu comme un support à l'évaluation, un outil de travail pour l'étude de l'ergonomie des logiciels. Comme l'indique A. Pollier, l'évaluation manuelle établie par l'ergonome est biaisée par sa propre expérience de l'interface étudiée [Pollier 91]. Par suite, un ergonome identifie en moyenne moins de la moitié des problèmes potentiels de l'interface évaluée. En conséquence, il s'avère intéressant de fournir à l'ergonome une aide "objective" en vue d'une évaluation plus performante.

8. Conclusion

Ce travail constitue un premier pas en direction de l'évaluation automatique de prototypes exécutables. Il vient en complément des modèles et outils d'évaluation prédictive. Nous avons automatisé la capture des actions physiques de l'utilisateur et réussi à les relier aux tâches élémentaires d'un arbre de séquences. Nous avons jeté l'esquisse d'une base de règles de comportement identifiées expérimentalement et nous avons montré comment les relier à l'interprétation ergonomique.

Il nous faut maintenant affiner la base de règles et son interprétation. En particulier, il est désirable que les patterns de comportement récurrents soient détectés de manière automatique. Ceci signifie que le système doit disposer d'un mécanisme d'apprentissage pour élaborer de nouvelles règles. De même les principes ergonomiques évoqués dans notre esquisse doivent être affinés. Par exemple, la cohérence revêt de multiples facettes parfois discutables [Grudin 89]. A cette fin, les travaux mi-formels de H. Thimbleby pourraient nous servir de base de travail [Thimbleby 90].

9. Bibliographie

[Balacheff 91]
N. Balacheff, "Contribution de la didactique et de l'épistémologie aux recherches en EIAO", Actes des XIIIèmes Journées Francophones sur l'Informatique, Genève, Janvier1991, pp. 9-38

[Barthet 88]
M.F. Barthet, "Logiciels interactifs et ergonomie", Editions Dunod Informatique, 1988

[Caelen 92]
J. Caelen & A.L. Fréchet, "Attitudes cognitives et actes de langage en situation de communication homme/machine", article soumis au congrès de l'ARC, Nancy 1992

[Card 83]
S.K. Card, T.P. Moran & A. Newell, "The psychology of Human Computer Interaction", Lawrence Erlbaum Associates, 1983

[Chabert 91]
A. Chabert, "La programmation visuelle", Rapport de DEA d'Informatique, Institut National Polytechnique de Grenoble, Juin 1991

[Coutaz 91]
J. Coutaz, "Architectural Design for User Interfaces", in ESEC'91, A.Van Lamsweerde & A. Fugetta (Eds), pp. 7-22, Milan, Italy, October 1991

[Cypher 91]
A. Cypher, "EAGER: Programming Repetitive Task by Example", Proceedings of CHI'91 (New Orleans, Apr. 28-May 2). ACM, New York, 1991, pp. 445-446.

[Grudin 89]
J. Grudin, "The Case Against User Interface Consistency", Communications of the ACM, 32(10), 1989, pp. 1164-1173

[Hardman 89]
B. Hardman Hoshstrasser, N. D. Geddes, "OPAL, Operator Intent Inferencing for Intelligent Operator Support Systems", Technical Report, Search Technology, Inc., 4725 Peachtree Corners Circle, Norcross, GA 30092

[Harrison 87]
M.D. Harrison & A.J. Dix, "Formalising Models of Interaction in the Design of a Display Editor", in Proceedings of the Second IFIP Conference on Human-Computer Interaction-INTERACT'87, pp. 409-414

[Hayes-Roth 79]
B. Hayes-Roth & F. Hayes-Roth, "A Cognitive Model of Planning", Cognitive Science 3, pp. 275-310, 1979

[Jacob 90]
R.J.K. Jacob, "What you look is what you get : eye mouvement-based interaction techniques", in Proceedings of CHI'90 conference, ACM Press, April 1990, pp. 11-18

[Kieras 85]
D. Kieras, P.G. Polson "An Approach to the Formal Analysis of User Complexity", International Journal of Man-Machine Studies, 22, 1985, pp. 365-394

[Lee 90]
K. F. LEE, R. REDDY, H. SIAO-WUEN, "An overview of the SPHINX speech recognition system", IEEE Trans. Acous. Speech, and Sig. Processing, 38(1), January, 1990, pp. 34-45

[Lerch 89]
F.J. Lerch, M.M. Mantei, J.R. Olson, "Skilled financial planning: The Cost of Translating Ideas into Acions", in Proceedings of CHI'89, ACM Press, 1989, pp. 121-126.

[Löwgren 90]
J. Löwgren & T. Nordqvist, "A Knowledge-Based Tool for User Interface Evaluation and its Integration in a UIMS", Human-Computer Interaction-INTERACT'90, pp. 395-400

[Moran 81]
T. Moran, ""The Command Langage Grammar : a representation for the user interface of interactive computer systems", International Journal of Man-Machine Studies, 15, 1981, pp. 3-50

[Norman 86]
D.A. Norman & S.W. Draper, "User Centered System Design", Lawrence Erlbaum Associates, 1986

[Normand 92]
V. Normand, "Le modèle Sirocco : de la spécification conceptuelle des interfaces utilisateur à leur réalisation", thèse de l'Université Joseph Fourier, 1992

[Pollier 91]
A. Pollier, "Evaluation d'une interface par des ergonomes : diagnostics et stratégies", Rapport de recherche INRIA no 1391, Février 1991

[Scapin 86]
D. L. Scapin, "Guide ergonomique de conception des interfaces homme-machine", Rapport Technique INRIA no 77, Octobre 1986

[Scapin 89]
C. Pierret-Goldreich, I. Delouis & D.L. Scapin, "Un Outil d'Acquisition et de Représentation des Taches Orienté-Objet", rapport de recherche INRIA no 1063, Programme 8 : Communication Homme-Machine, août 1989

[Terzopoulos 90]
D. Terzopoulos & K. Waters, "Analysis of facial images using physical and anatomical models", in IEEE, 1990, pp. 727-732

[Thimbleby 90]
H. Thimbleby, "User Interface Design", ACM Press, New York, Frontier Series, Addison-Wesley Publ., 1990

[Yuille 90]
A.L. Yuille, "Deformable templates for face recognition", in Journal of Cognitive Neuroscience, 3(1), 1991, pp. 59-70

[Young 90]
R.M. Young & J. Whittington, "Interim Report on the Instruction Langage", AMODEUS Project Document: Deliverable D5, ESPRIT Basic Research Action 3066, 11th August 1990