Augmenter la validité des évaluations des applications graphiques interactives

Caroline Appert

Augmenter la validité des évaluations des applications graphiques interactives

Combiner les approches prédictive et empirique

Caroline Appert

Laboratoire de Recherche en Informatique
Université Paris-Sud Orsay et CNRS, F-91405 Orsay Cedex
appert.lri@fr

RÉSUMÉ. *Alors que la recherche en interaction homme-machine (IHM) produit nombre de techniques d'interaction pour améliorer l'utilisabilité des applications graphiques, la plupart de nos interfaces restent des assemblages selon le modèle WIMP (Windows, Icons, Menus and Pointing). Ce décalage peut s'expliquer en partie par la difficulté pour les développeurs d'évaluer les gains de performance offerts par une technique dans un contexte réel. Afin d'améliorer cet état de fait, cet article propose une méthodologie d'évaluation combinant des aspects empiriques et prédictifs pour augmenter la validité externe des résultats rapportés dans la littérature en IHM. Cette approche repose sur deux outils, le modèle CIS (Complexity of Interaction Sequences) et la plateforme TouchStone, et a pour but de permettre aux développeurs d'interfaces graphiques l'exploration d'un espace de conception dans lequel faire des choix informés.*

ABSTRACT. *While human-computer interaction (HCI) researchers invent a lot of interaction techniques that could improve usability of graphical applications, most of our interfaces are still based on the WIMP model (Windows, Icons, Menus and Pointing). The difficulty of assessing the performance of an interaction technique in a real context of use can partly explain this gap. This article aims at improving the evaluation process used in HCI by following a methodology where predictive and empirical aspects are complementary to augment the external validity of lab experiments. This approach is supported by two tools: the Complexity of Interaction Sequences (CIS) model and the TouchStone platform. This evaluation methodology defines a design space with performance measures to allow interface developers to make informed choices.*

MOTS-CLÉS : *interaction, évaluation, expérimentation, contexte d'utilisation, validité.*

KEYWORDS: *interaction, evaluation, experiment, context of use, validity.*

1. Introduction

L'interaction homme-machine (IHM) considère l'interaction entre l'utilisateur et un système informatique comme un phénomène à étudier afin de mieux le comprendre et de l'améliorer. Plus spécifique que l'ergonomie qui vise à apprécier confort et performance d'un système entre un homme et un artefact, l'IHM se concentre sur les systèmes informatiques. Pour l'évaluation des performances de nouvelles techniques d'interaction, les chercheurs en IHM s'inspirent des méthodes de la psychologie expérimentale qui mesure les réponses de l'être humain à des stimuli dans son environnement physique. Cependant, le passage à l'échelle de ces méthodes vers le monde virtuel montre de nouveaux défis. En effet, ici, un des buts de la recherche en IHM est d'offrir de nouveaux moyens pour l'être humain d'exprimer ces réponses. La complexité des protocoles expérimentaux se trouve donc augmentée par la multiplication des conditions à considérer.

Ainsi, la littérature en IHM propose un grand nombre de résultats empiriques mais ne comparant bien souvent qu'un sous-ensemble des techniques possibles et sur une tâche bien spécifique. C'est peut-être ici que se situe un des verrous entre les innovations de la recherche et les applications interactives courantes. En effet, depuis 1981 et l'interface du Xerox Star (Smith *et al.*, 1986) (figure 1), les interactions disponibles dans les applications graphiques actuelles sont principalement restées dans des principes WIMP (Windows, Icons, Menus and Pointing) alors que la recherche en IHM produit continuellement de nouvelles techniques d'interaction pour améliorer l'utilisabilité des applications graphiques. Par exemple, le temps de sélection d'une commande peut être trois fois plus rapide avec un menu circulaire (Callahan *et al.*, 1988) qu'avec les habituels menus linéaires. Ou encore, la toolglass (Bier *et al.*, 1993) est une palette d'outils semi-transparente que l'utilisateur peut déplacer à l'aide d'un second périphérique d'entrée. Au lieu d'avoir deux actions séquentielles pour sélectionner l'outil puis l'appliquer sur l'objet d'intérêt, l'utilisateur peut utiliser ses deux mains pour déplacer palette d'outils et curseur vers l'objet d'intérêt et cliquer sur l'objet au travers de l'outil pour l'appliquer. Ici encore, les résultats expérimentaux rapportent un gain en temps qui peut atteindre 40 % en comparaison avec l'utilisation d'une traditionnelle palette d'outils.

L'introduction de techniques innovantes dans les applications courantes ne peut se faire que si les concepteurs et développeurs d'interfaces sont en mesure d'*évaluer* la performance de ces techniques et de les *intégrer* aux applications interactives. Sans minorer les obstacles lors de l'intégration aux applications (certaines techniques sont brevetées, requièrent une configuration matérielle spécifique ou encore sont difficiles à programmer avec les outils de développement actuels), cet article se concentre sur les problèmes liés à l'évaluation des techniques d'interaction. Il propose une méthodologie dans laquelle les approches empirique et prédictive se complètent pour augmenter la validité interne et externe des résultats expérimentaux. Cette méthodologie repose sur deux outils : le modèle Complexity of Interaction Sequences (CIS) qui permet d'établir le lien entre performance d'une technique d'interaction et contexte d'utilisation et la plateforme TouchStone qui permet de rationnaliser et capitaliser les

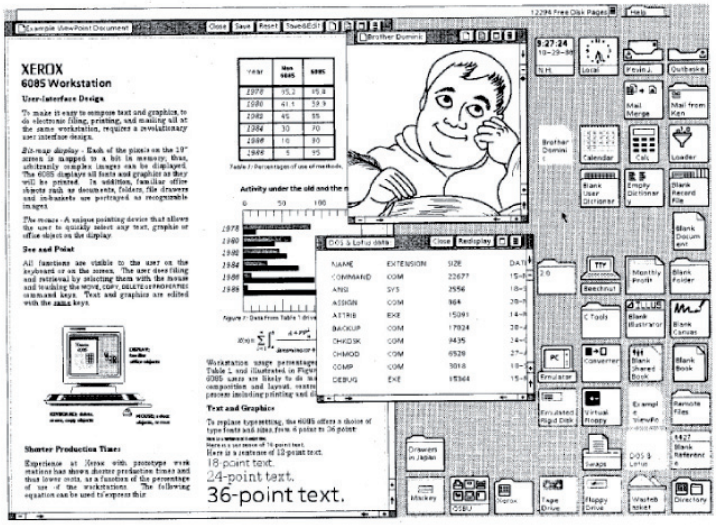

Figure 1. *L'interface du Xerox Star*

évaluations empiriques afin de proposer des résultats qui passent à l'échelle malgré la multiplication des conditions.

2. Évaluation des nouvelles techniques d'interaction

2.1. *Expérimentations en laboratoire*

La manière la plus courante de valider une nouvelle technique d'interaction est son évaluation en laboratoire. Dans l'impossibilité de réaliser des expériences exhaustives, les expérimentateurs sélectionnent une tâche expérimentale et un échantillon de techniques d'interaction concurrentes. La mesure de performance la plus souvent utilisée est le temps d'exécution de la tâche. Ainsi, une technique est considérée comme « meilleure » si elle permet d'accomplir plus rapidement cette tâche spécifique. La validité externe des résultats reportés repose donc bien évidemment sur la représentativité de la tâche par rapport à un contexte d'utilisation réel et l'échantillon de techniques choisi par rapport à l'ensemble de techniques existantes. (MacKenzie *et al.*, 2001) montrent que, malgré l'abondance des publications en IHM et dans le domaine des facteurs humains, les méthodologies sont *ad hoc* et les différentes procédures expérimentales sont bien souvent inconsistantes entre elles.

Rares sont les propositions pour améliorer cet état de fait. Le Generalized Fitts' Law Model Builder (Soukoreff *et al.*, 1995) ou la plateforme Shakespeare (Guiard *et al.*, 2006) sont tous les deux des systèmes exclusivement dédiés à la réalisation d'ex-

périmentations d'un type très spécifique : l'évaluation d'une technique de pointage selon le protocole expérimental proposé par (Fitts, 1954). Ces outils sont très utiles, par exemple, pour mener des expérimentations dans le contexte du standard ISO 9241 (Soukoreff *et al.*, 2004) sur les dispositifs physiques de pointage (souris, stylet, etc.) et les améliorations logicielles (grossisement des cibles, réduction des distances, etc.) mais ne permettent pas de couvrir l'ensemble des expérimentations existantes. Récemment, (Gray *et al.*, 2004) ont proposé un système pour faciliter le développement de variantes de techniques d'interaction pour les comparer en séparant clairement la vue du modèle et en permettant de spécifier une vue qui ne couvre pas tout le modèle (pour comparer des vues qui affichent plus ou moins d'informations par exemple). Ici encore, ce système montre également des limites puisqu'il ne couvre que les interactions sur dispositifs portables.

2.2. *Modèles de l'interaction*

Puisqu'il semble difficile de couvrir l'ensemble des possibles par des évaluations empiriques, une alternative réside en la modélisation de l'interaction afin de prédire les performances. La famille de modèles prédictifs GOMS (*Goals, Operators, Methods and Selection rules*) (John *et al.*, 1996), basés sur l'analyse de tâches, est incontestablement le travail le plus significatif dans ce sens. CMN-GOMS, NGOMSL et CPM-GOMS décrivent une tâche comme une hiérarchie de buts à atteindre avec des opérateurs aux feuilles. Un but peut être atteint par une méthode composée de sous-buts et d'opérateurs. *Keystroke-Level Model* (KLM) est l'exception dans cette famille de modèles et décrit une tâche comme une simple séquence ordonnée d'opérateurs sans structure hiérarchique. À l'exception de KLM, ces modèles de haut niveau sont, d'une part, coûteux en temps de modélisation et, d'autre part, en réduisant l'interaction à une séquence ordonnée d'opérateurs ils ne permettent pas d'exhiber explicitement les propriétés des techniques et leur lien avec le contexte d'utilisation.

D'autre modèles, non prédictifs, décrivent au contraire l'interaction en termes de propriétés ou de dimensions. Par exemple, (Card *et al.*, 1991) proposent un espace de conception pour les périphériques d'entrée. Dans ce dernier modèle, un périphérique d'entrée est défini comme un traducteur d'actions physiques en commandes logiques de bas niveau, il peut être placé dans un espace composé de dimensions descriptives (ex : relatif *vs.* absolu, linéaire *vs.* rotatif, etc.) et décrit comme le résultat de l'application d'un opérateur sur ces dimensions. Par exemple, la souris inclut une composition de deux axes relatifs linéaires pour former un dispositif de pointage en deux dimensions. Les différentes dimensions descriptives forment un espace de conception, un périphérique étant une composition d'un ensemble de points dans cet espace. Cependant, le périphérique d'entrée n'est qu'une vue incomplète du phénomène d'interaction qui implique entre autres des objets graphiques de l'application. Le modèle de l'interaction instrumentale (Beaudouin-Lafon, 2000) offre un niveau d'abstraction plus élevé et propose de penser l'interaction en termes d'objets du domaine et d'instruments pour agir sur ces objets du domaine. L'instrument transforme les actions de l'utilisateur en

commandes qui agissent sur les objets du domaine. L'instrument réagit aux actions de l'utilisateur et fournit un feedback quant à l'état de l'objet qu'il contrôle. L'interaction instrumentale est un modèle peu formel qui permet de comparer des instruments selon un ensemble de propriétés faciles à utiliser. Par exemple, le *degré d'indirection* mesure la distance spatiale entre objet et instrument et distance temporelle entre l'action de l'utilisateur et l'effet sur l'objet. Par exemple, une barre de défilement a un degré d'indirection plus fort que l' « outil main » des applications Adobe qui permet de faire glisser un document en l'« attrapant » directement. Le niveau d'abstraction de l'interaction instrumentale permet de laisser des espaces de conception ouverts et fournit des principes pour l'explorer. Par exemple, le principe de réification consiste à transformer des concepts en objets concrets. Dans une perspective d'évaluation, la caractéristique « générative et exploratoire » des deux derniers modèles évoqués est très intéressante car elle permet de lister les différentes alternatives et ainsi évaluer un échantillon représentatif des techniques permettant d'exécuter la même opération. Cependant, la formalisation de l'interaction est ici trop abstraite pour fournir des prédictions quant à la performance de l'interaction décrite.

3. Établir le lien entre performance et contexte d'utilisation avec CIS

La modélisation de l'interaction permet donc d'offrir une perspective particulière pour l'analyse du phénomène étudié et, dans certains cas, de prédire la performance des utilisateurs. Cette section présente le modèle *Complexity of Interaction Sequences*, CIS (Appert *et al.*, 2004), qui permet de décrire différentes techniques d'interaction pour activer les commandes de l'application. À l'inverse des modèles existants, le niveau d'abstraction n'est ni trop élevé ni trop bas pour, d'une part, pouvoir analyser les techniques selon un ensemble de propriétés de haut niveau favorables à la génération et l'exploration et, d'autre part, s'appuyer sur des lois empiriques permettant de prédire leur efficacité. Une originalité de CIS est le lien entre performance et contexte d'utilisation, les prédictions étant toujours liées à une séquence d'interaction qui opérationnalise un contexte d'utilisation réel.

3.1. *Décrire la structure de l'interaction avec CIS*

Dans CIS, une technique d'interaction est décrite sous la forme d'un graphe orienté sans cycle dans lequel un chemin de la racine à une des feuilles décrit les actions sensori-motrices que l'utilisateur doit faire pour déclencher une commande de l'application. Ainsi, CIS considère l'ensemble des commandes de l'application comme l'espace d'interaction et une action sensori-motrice comme un filtre sur cet ensemble. Un graphe CIS modélise la façon dont ces actions doivent être appliquées afin de réduire l'espace d'interaction à une unique commande qui est alors déclenchée.

La figure 2 montre un exemple simple : le graphe CIS d'une palette d'outils traditionnelle pour un éditeur graphique qui permet de créer des triangles, ellipses et

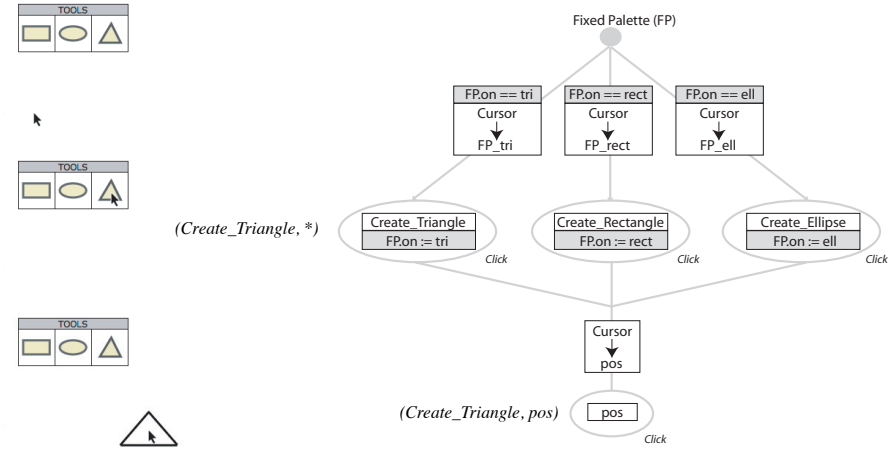

Figure 2. *Le graphe CIS d'une palette d'outils*

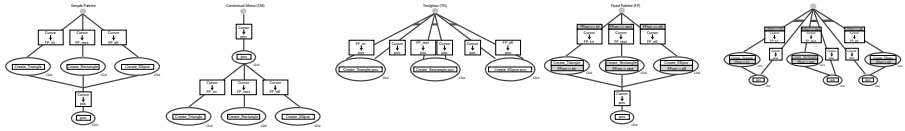

Figure 3. *Différentes structures de graphe CIS pour activer une même commande*

rectangles de taille prédéfinie. L'espace d'interaction est alors l'ensemble des commandes de la forme :

$$(t \in \{Create\_Triangle,\ Create\_Rectangle,\ Create\_Ellipse\},\ p \in position)$$

Pour comprendre la signification des différents éléments d'un graphe, considérons le chemin le plus à gauche. Le branchement à la racine du graphe traduit le choix entre les trois outils de la palette. Le premier arc décrit le déplacement du curseur vers l'outil pour créer des triangles et désigne donc le sous-espace d'interaction $(Create\_Triangle,\ *)$. Le nœud sortant de cet arc représente l'action ponctuelle (un clic par exemple) qui valide cette réduction de l'espace d'interaction. S'en suit le pointage de la position à laquelle le triangle doit être créé et le clic de validation qui réduit l'espace d'interaction à la seule commande $(Create\_Triangle,\ pos)$. Un triangle est alors créé à la position pos et l'état de l'interface s'en trouve modifié.

La description d'une technique sous forme de graphe CIS permet d'exhiber sa structure morphologique. Ce niveau d'analyse de l'interaction est propice à la *génération* de techniques alternatives à l'image des modèles non prédictifs mention-

nés dans la section précédente. En effet, la structure d'un graphe peut être changée par différentes opérations afin d'envisager d'autres graphes déclenchant les mêmes commandes. Ces opérations consistent à changer l'ordre des actions, paralléliser des actions, conférer de la persistance à certaines actions, développer, factoriser ou fusionner des actions. Le concepteur peut ainsi envisager des alternatives en considérant les différentes structures de graphe possibles. Par exemple, en partant de l'exemple de la palette fixe, on peut paralléliser les actions d'acquisition de l'outil et de l'objet d'intérêt et obtenir ainsi une toolglass, changer l'ordre des actions « outil-objet » en « objet-outil » comme dans le cas d'un menu contextuel, etc. CIS constitue ainsi un guide quant à l'exploration des différentes alternatives à envisager pour déclencher la même commande.

3.2. *Prédire l'efficacité en contexte avec CIS*

Une fois décrite sous forme de graphe CIS, il est possible de prédire la performance d'une technique. Les prédictions de CIS s'appuient sur les lois prédictives sous-jacentes à l'interaction des applications graphiques. Les trois lois les plus connues et les plus utilisées en IHM sont respectivement les lois de Fitts, de Hick-Hyman et de « steering-path ». La loi de Fitts (Fitts, 1954) évalue la difficulté et le temps de mouvement pour le *pointage* d'une cible. La loi de Hick-Hyman (Hick, 1952; Hyman, 1953) évalue le temps de *choix* d'un élément dans un ensemble en fonction du nombre d'éléments de l'ensemble. La loi de steering-path (Accot *et al.*, 1997), plus récente, évalue le temps de *suivi d'une trajectoire*. Ces trois lois réunies permettent de couvrir la majorité des interactions dans une application graphique et motivent le niveau d'abstraction des briques de base, ou *primitives de l'interaction*, du modèle CIS.

Ces lois permettent de prédire le temps nécessaire à une action CIS et par conséquent le temps nécessaire à l'activation d'une commande C en construisant le chemin dans le graphe qui réduit l'espace des commandes à l'unique commande C. C'est le principe sur lequel repose le simulateur SimCIS qui prédit la performance d'une technique pour un contexte d'utilisation à partir du graphe de la technique et d'une séquence de commandes opérationnalisant le contexte. En effet, dans l'idée d'atteindre un but, défini comme un état de l'interface à atteindre, l'utilisateur peut choisir différentes techniques d'interaction et différentes séquences d'interaction. Ces choix sont guidés par l'état courant de l'interface, les techniques qui sont à sa disposition, la connaissance qu'il a de cette interface et la quantité de commandes qu'il est en mesure de planifier. Certaines activités (au sens de (Green, 2000)), comme la recopie, demandent une faible charge cognitive et permettent donc aux utilisateurs de planifier longtemps en amont leurs futures interactions alors que d'autres, comme la résolution de problème, sont plus difficiles et les utilisateurs les abordent de manière plus incrémentale autour d'un point d'intérêt (Mackay, 2002). Nous définissons le contexte d'utilisation comme la combinaison de l'état courant de l'interface et la quantité de planification que l'utilisateur peut faire. En termes CIS, ceci se traduit par une séquence de commandes appelée *séquence d'interaction*. L'état courant de l'interface

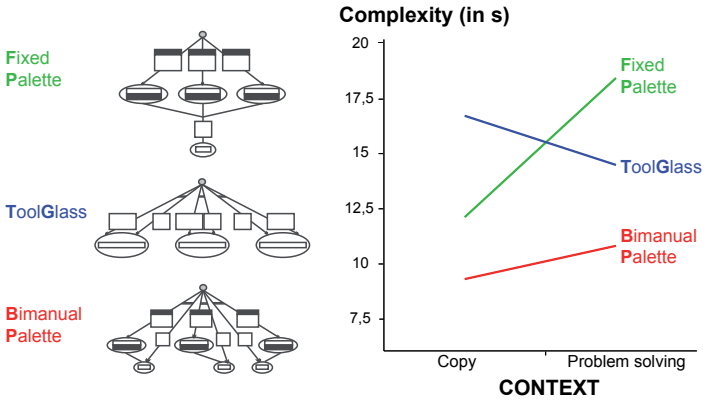

Figure 4. *Les complexités en temps de trois techniques (palette fixe, toolglass et palette bi-manuelle) pour deux contextes d'utilisation (recopie ou résolution de problème)*

dépend des interactions précédentes alors que la planification des interactions futures dépend du contexte d'utilisation. C'est en considérant les performances d'une technique sur une séquence d'interaction représentative d'un type d'activité que CIS capture une partie du contexte dans ses prédictions.

La figure 4 reporte les prédictions de performance pour trois techniques différentes dans deux contextes d'utilisation (tâche de recopie ou tâche de résolution de problème). On peut en particulier observer que les performances relatives entre deux techniques s'inversent selon le contexte d'utilisation, confirmant ainsi le lien étroit qui existe entre l'efficacité d'une technique et le contexte cognitif dans lequel elle est utilisée. Ce résultat soutient également l'intérêt d'envisager différentes structures de graphe : la structure hybride de la palette bi-manuelle qui allie les propriétés de persistance de la palette fixe et de parallélisme de la toolglass la rend non seulement plus performante dans les deux contextes mais aussi moins sensible au contexte. C'est ici qu'intervient le parallèle avec la notion de complexité utilisée en algorithmique qui est une des originalités de CIS. De la même façon qu'un algorithme exhibe une complexité variable selon la nature des données fournies en entrée, une technique d'interaction présente des performances variables en fonction du contexte d'utilisation.

Afin de valider CIS, nous avons mené une expérimentation contrôlée qui confronte les prédictions de CIS à des observations empiriques. Cette expérimentation compare trois techniques ayant des structures CIS différentes sur quatre contextes d'utilisation et montre que les différences empiriques entre techniques et entre contextes sont les mêmes que les différences prédites (Appert *et al.*, 2004). Ces résultats illustrent également le biais qui peut exister en fonction de la tâche expérimentale choisie. Dans (Kabbash *et al.*, 1994), les toolglasses sont comparées aux palettes et sont rapportées plus efficaces que les palettes. Ce résultat n'est pas surprenant si on considère

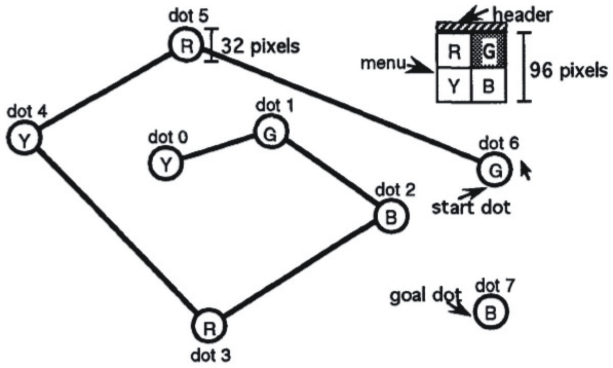

Figure 5. *La tâche « connect the dots » (Illustration extraite de (Kabbash* et al.*, 1994))*

finement la tâche qui a été utilisée pour comparer ces techniques. Dans la tâche de Kabbash *et al.*, « connect the dots » (figure 5), des points colorés apparaissent l'un après l'autre et l'utilisateur doit sélectionner la couleur du nouveau point avant de le relier par drag'n drop au point précédent. Dans (Kabbash *et al.*, 1994), deux points successifs ont toujours deux couleurs différentes, ce qui est proche de l'opérationnalisation d'un contexte de résolution de problème dans lequel l'utilisateur doit changer d'outil à chaque fois.

3.3. *Complémentarité des approches prédictive et empirique*

Ne modélisant que les aspects sensori-moteurs de l'interaction, CIS n'a pas pour but de remplacer les évaluations empiriques qui permettent de capturer l'ensemble du phénomène étudié. Cependant, les expérimentations menées en laboratoire ne considèrent que des cas spécifiques puisqu'il faut choisir une tâche expérimentale. CIS permet au contraire d'évaluer facilement les variations de performance d'un contexte à l'autre. Au niveau d'une séquence d'interaction, CIS vient donc *renforcer la validité* d'une évaluation empirique en évaluant la représentativité de la tâche expérimentale parmi les différents contextes d'utilisation possibles.

Aussi, la définition du modèle CIS demande un découpage en primitives de l'interaction (qui correspondent aux actions CIS) et la connaissance des lois prédictives sous-jacentes à ces différentes primitives. La découverte de telles lois prédictives se fait expérimentalement pour valider une caractérisation du phénomène par un ensemble de variables abstraites (c'est-à-dire indépendantes du domaine) explicatives des performances de l'utilisateur. Au niveau de la primitive d'interaction, CIS *repose* donc sur des modèles élaborés empiriquement.

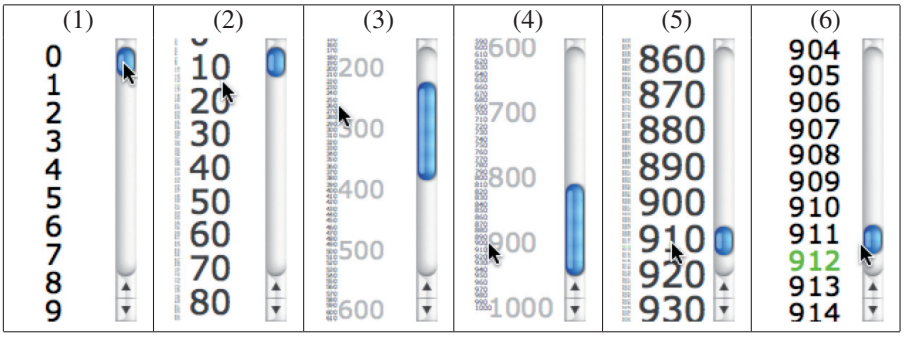

Figure 6. *OrthoZoom*

4. Augmenter la couverture des évaluations empiriques

Cette section commence par l'illustration du problème de *couverture partielle* que posent les évaluations couramment menées en IHM en s'appuyant sur l'exemple de l'introduction d'une nouvelle primitive de pointage[1], OrthoZoom. Dans un second temps, elle présente la plateforme TouchStone qui permet d'augmenter cette couverture et donc la validité des résultats empiriques.

4.1. *Introduction d'une nouvelle interaction*

OrthoZoom (Appert *et al.*, 2006), illustré par la figure 6, est une primitive de *pointage* pour atteindre rapidement un élément dans un très grand document linéaire à l'aide d'un simple périphérique de pointage (comme une souris ou un stylet). OrthoZoom augmente une traditionnelle barre de défilement en attribuant le degré de liberté du périphérique, habituellement ignoré, au contrôle du facteur d'échelle. Ainsi, les déplacements le long de la dimension du document contrôlent la position du document (comme avec une barre de défilement traditionnelle) alors que les déplacements le long de la dimension orthogonale contrôlent le facteur de zoom. Ainsi, une simple trajectoire courbe à la souris permet de réaliser rapidement le déplacement en 3 phases observé par (Guiard *et al.*, 1999) dans leur étude sur le pointage multi-échelle. Dans ce type d'interfaces, augmentées d'une dimension de zoom, la trajectoire la plus courte d'un point à un autre n'est pas la ligne droite mais la séquence (i) un zoom arrière, (ii) un léger déplacement puis (iii) une séquence de zoom avant avec des corrections pour atteindre la cible.

Étant donné le grand nombre de techniques concurrentes existantes, OrthoZoom a été comparé à la technique la plus efficace connue précédemment et qui *repose sur*

1. Un pointage correspond à un arc CIS.

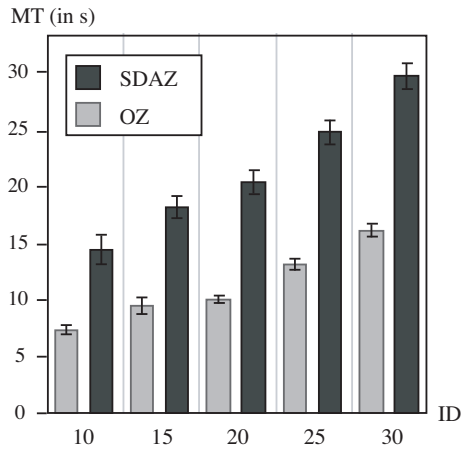

Figure 7. *Temps de mouvement (MT) moyen en fonction de la difficulté de pointage pour les techniques OrthoZoom et Speed Dependant Automatic Zooming*

la même configuration d'entrée (Appert *et al.*, 2006). En effet, des techniques concurrentes comme le pointage bimanuel (Guiard *et al.*, 1999) qui utilise deux périphériques d'entrée ou Zliding (Ramos *et al.*, 2005) qui utilise la pression du stylet sont des techniques efficaces mais qui sont bien souvent impossibles à mettre en œuvre dans les configurations matérielles habituelles. A l'inverse, Speed Dependant Automatic Zooming (SDAZ) (Igarashi *et al.*, 2000) propose une simple interaction de « drag » avec un périphérique de pointage standard comme OrthoZoom. La différence avec OrthoZoom est qu'elle ne laisse pas le contrôle direct du niveau de zoom à l'utilisateur mais l'ajuste automatiquement en fonction de la vitesse de défilement. Le vecteur vitesse est déterminé par le point initial et le point courant de l'interaction de drag, et le facteur de zoom est fonction de ce vecteur (plus le déplacement est rapide, plus l'échelle du document est grande). En 2003, (Cockburn *et al.*, 2003) avaient montré que SDAZ était la technique la plus efficace pour se déplacer dans de grands documents linéaires avec une souris.

La figure 7 montre les principaux résultats de la comparaison empirique des performances d'OrthoZoom (OZ) et de Speed Dependant Automatic Zooming (SDAZ) sur la tâche expérimentale de référence pour le pointage. Le graphe de gauche rapporte le temps moyen de pointage en fonction de la difficulté de pointage. On peut voir qu'OZ est en moyenne 2 fois plus rapide que SDAZ, y compris pour des pointages d'une extrême difficulté (jusqu'à 30 bits). La difficulté d'un pointage peut s'exprimer en termes de quantité d'information à transmettre au système pour identifier un élément parmi l'ensemble des éléments disponibles. À titre d'illustration, désigner une base azotée parmi les 3 milliards de bases azotées du génome humain représente une difficulté de $\sim$31 bits.

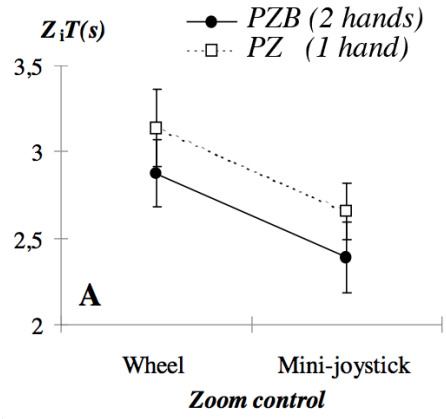

Figure 8. *Temps de Zoom Control (ZT) avec une technique de Pan/Zoom bi-manuel (PZB) ou une technique de Pan/Zoom uni-manuel (PZ) (Illustration extraite de (Bourgeois* et al.*, 2001))*

4.2. *Méthodologie pour l'évaluation empirique*

L'évaluation rapportée ci-avant a permis de montrer qu'OrthoZoom est la technique la plus performante pour des tâches de pointage en 1D avec un périphérique de pointage 2D standard. Cependant, ces résultats ne permettent pas de savoir quelle technique est la plus efficace parmi l'ensemble des techniques existantes sans imposer de contraintes sur la configuration d'entrée (souris augmentée d'une molette, configuration bi-manuelle impliquant deux périphériques d'entrée, etc.). La réalisation d'expériences exhaustives est coûteuse pour les chercheurs en IHM pour entre autres développer chacune des conditions et l'introduire dans le plan expérimental. Ainsi, les concepteurs d'interfaces se trouvent face à des ensembles de résultats incomparables. Par exemple, d'un côté ils peuvent apprécier qu'OZ est plus efficace que SDAZ pour une tâche de *Zoom Control* de pointage 1D et de l'autre côté qu'une configuration bi-manuelle est plus efficace qu'une configuration uni-manuelle sur la phase finale de pointage (qui consiste en une succession d'opérations de zoom avant avec des corrections pour « atterrir » sur la cible). En effet, les résultats de l'étude (Bourgeois *et al.*, 2001) illustrée par la figure 8 montre la supériorité de l'interaction bi-manuelle par rapport à l'interaction aussi bien avec une souris à molette (*Wheel*) qu'avec un joystick (*Mini-joystick*) pour contrôler le facteur de zoom.

La plateforme TouchStone (Mackay *et al.*, 2007) développée par l'équipe in|situ| permet, grâce à (i) la définition d'un format d'expérimentation, (ii) une architecture modulaire et (iii) une fonction d'entrepôt, de partager des composants d'expérimentation afin de permettre non seulement la réplication mais aussi l'extension de protocoles expérimentaux à moindre coût. Ainsi, il est possible de facilement généraliser les ré-

sultats de l'étude de la figure 8 et de celle de la figure 7 pour connaître la technique la plus efficace pour une tâche de pointage sans contrainte sur la configuration d'entrée. La section suivante présente la plateforme d'évaluation TouchStone et comment les résultats d'études menées à l'aide de cette plateforme peuvent être capitalisés pour obtenir de nouveaux résultats à moindre coût.

4.3. *La plateforme TouchStone*

TouchStone comprend trois plateformes pour couvrir les trois principales parties que comporte la réalisation d'une expérimentation contrôlée : une plateforme de conception, une plateforme d'exécution et une plateforme d'analyse.

La plateforme de conception est une application internet qui découpe la construction d'un plan expérimental en différentes étapes structurées. Lors de ces étapes (spécification des facteurs, de la stratégie de contrebalancement, estimation du temps nécessaire, etc.), l'évaluateur peut tester plusieurs alternatives afin de faire le choix qui lui convient le mieux. Afin d'encourager une approche exploratoire du plan expérimental, l'application permet également à l'évaluateur de revenir sur n'importe quelle étape à chaque instant. La plateforme de conception permet de produire un fichier XML qui décrit l'expérimentation en termes d'instructions séquentielles « exécutables » tout en étant lisibles par un humain. Ce script définit l'enchaînement des différentes conditions et référence les composants logiciels nécessaires à l'exécution de l'expérimentation. La plateforme d'exécution interprète ce script pour l'exécuter et produire des fichiers d'enregistrements destinés à être analysés avec la plateforme d'analyse[2]. L'objet de cet article n'est pas de fournir une description complète de TouchStone (voir (Mackay *et al.*, 2007) et (Appert, 2007) pour plus de détails) mais d'illustrer l'architecture modulaire qui permet de « brancher » facilement une nouvelle condition dans un plan expérimental.

La plateforme d'exécution[3] comprend un *moteur d'expérimentation* qui automatise la partie commune à toute expérimentation et une *interface de développement* (API) pour développer les composants logiciels spécifiques à une expérimentation donnée. Une expérimentation TouchStone est formée du couple (protocole expérimental, composants logiciels d'exécution), c'est-à-dire non seulement du script XML généré par la plateforme de conception mais aussi des *composants logiciels* référencés dans ce script et développés avec la plateforme d'exécution. Dès lors, les buts de la plateforme d'exécution sont :

– un *moteur d'expérimentation* qui offre la plus grande modularité possible et automatise la plus grande partie commune à toute expérimentation pour minimiser le

2. TouchStone est un projet encore en cours de développement. La plateforme d'analyse est, pour l'instant, un site internet fournissant quelques conseils sur les analyses possibles et les logiciels d'analyse couramment utilisés.
3. La plateforme d'exécution de TouchStone est un projet open source (http ://gforge.inria.fr/projects/multiscalenav/) contenant 13 000 lignes de code Java.

```
<interblock class="Message({New block})" criterion="Key(Space)">
   <block class="PointingBlock" values="T=OZ"
          criterionTrial="Dwell(1000) | (TimeOut(180000)=>{Too Long})" >
      <intertrial class= "Message({Touch the target as quickly as possible!})"
                  criterion="Press(Mouse.Left)">
      <trial values="ID=15, W=400" class="PointingBlock" />
      <trial ... />
      <trial ... />
      ...
      </intertrial>
   </block>
   <block ... >
      ...
   </block>
</interblock>
```

Figure 9. *Extrait de script expérimental*

temps de programmation des évaluateurs finaux et faciliter la réutilisation de *composants logiciels* par plusieurs protocoles,

– pouvoir construire les composants logiciels d'expérimentation exécutables à partir d'un simple identifiant dans un format textuel comme XML.

Le script XML produit par la plateforme de conception décrit le protocole expérimental par une séquence de balises XML spécifiques (figure 9). La première partie du script déclare les différents facteurs et mesures de l'expérimentation tandis que la seconde partie spécifie l'exécution séquentielle (voir extrait ci-après) des essais expérimentaux. L'expérimentation est une séquence de blocs (balises `<block ...>`) qui peut être entrecoupée d'intertitres (balises `<interblock ...>`), chaque bloc étant une séquence d'essais (balises `<trial ...>`) qui peut également être entrecoupée d'intertitres (balises `<intertrial ...>`). Les attributs `values` spécifient les valeurs des facteurs pour chaque essai tandis que les attributs `criterion` (ou `criterionTrial`) spécifient les critères d'arrêt des intertitres (ou des essais). Les valeurs des attributs `class`, `values`, `criterion` et `criteriontrial` sont interprétées comme de simples chaînes de caractères sauf si l'une des fabriques de la plateforme associe un composant logiciel à cette chaîne de caractères, comme décrit ci-après.

Le cœur commun à toutes les expérimentations menées avec TouchStone est programmé dans le *moteur d'expérimentation* en utilisant la machine à états de la figure 10. Cette machine contient deux types d'états, les états *script* et les états *participant*, afin de gérer les deux types d'entrée du moteur d'expérimentation : les instructions du script XML et les actions du participant. L'entrée dans un état script reprend l'exécution du protocole expérimental et écoute les événements qu'il génère (c'est-à-dire les balises XML) alors que l'entrée dans un état participant bloque l'exécution du script jusqu'à ce qu'un événement *done* soit reçu. Cet évènement *done* est automatiquement envoyé dès lors que les actions du participant ont satisfait le critère spécifié dans le script XML. Le format des enregistrements produit par le moteur d'expérimentation respecte le standard des logiciels d'analyse et la vie privée des participants en

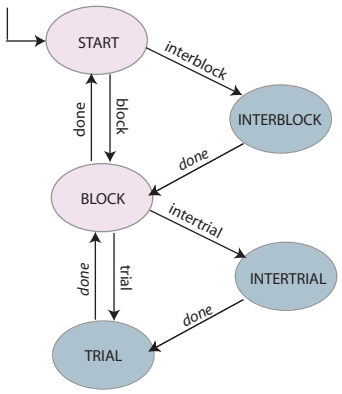

Figure 10. *Machine à états pour les deux sources d'événement d'une expérimentation*

utilisant des codes générés automatiquement pour représenter les participants dans les enregistrements destinés à l'analyse alors que les associations (code, participant) sont stockées dans un fichier indépendant destiné aux seuls évaluateurs pour le déroulement de l'expérience. Les limites du moteur d'expérimentation ont été choisies de manière à automatiser le maximum de traitements sans pour autant réduire la gamme d'expérimentations envisageables en IHM. A l'inverse des systèmes existants, TouchStone permet de réaliser des expérimentations sur le pointage mais aussi sur les interactions basées sur la reconnaissance de gestes ou encore sur les sélections de commandes en utilisant différents types de menu.

Afin que les composants logiciels d'une expérimentation puissent être référencés depuis le script XML par une simple chaîne de caractères, la plateforme d'exécution repose sur un mécanisme de fabriques de composants (*factory design pattern* (Gamma *et al.*, 1995)) couplé avec un *doclet*. D'une part, une fabrique est une classe Java qui gère une table d'association (*identifiant*, *classe Java*) et qui construit des objets java à la demande à partir d'un identifiant. D'autre part, le doclet de TouchStone permet d'associer un identifiant à une classe Java de manière déclarative. Cette déclaration est utilisée pour le référencement de ce composant logiciel par la plateforme de conception alors que la fabrique sera utilisée pour l'exécution dynamique de l'expérimentation en fabriquant le composant logiciel au moment où il est référencé par son identifiant dans le script XML. Par exemple, la classe Java `PBlock` ci-après est référençable par l'identifiant `PointingBlock` utilisé dans le script de la figure 9.

```
/**
  * touchstone.block PointingBlock
  **/
public class PBlock extends Block {
    ...
}
```

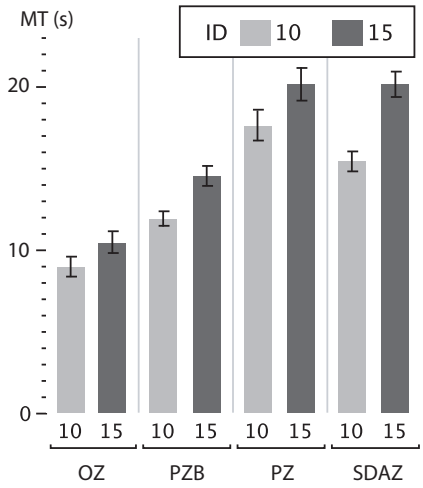

Figure 11. *Temps de mouvement (MT) moyen en fonction de la difficulté de pointage pour les techniques OrthoZoom (OZ), Pan/Zoom bi-manuel (PZB), Pan/Zoom uni-manuel (PZ) et Speed Dependant Automatic Zooming (SDAZ)*

L'architecture modulaire et les mécanismes de référencement de TouchStone permettent effectivement d'augmenter la validité des évaluations empiriques. Par exemple, OrthoZoom (OZ) a été comparé à Speed Dependant Automatic Zooming (SDAZ), la technique multi-échelle rapportée la plus efficace qui n'utilise qu'une souris, tandis que (Bourgeois *et al.*, 2001) avaient auparavant comparé le « pan and zoom » bimanuel (PZB) au « pan and zoom » unimanuel (PZ) qui demandent tous les deux une configuration d'entrée plus élaborée : un deuxième périphérique d'entrée (PZB) ou une souris augmentée d'une molette (PZ). Les deux expériences OZ *vs.* SDAZ et PZ *vs.* PZB ayant été réalisées dans des conditions différentes, il est impossible de déterminer laquelle choisir pour obtenir l'interface la plus performante sans contraintes sur la configuration d'entrée. Dans (Mackay *et al.*, 2007), nous montrons que si les deux expériences OZ *vs.* SDAZ et PZ *vs.* PZB ont été réalisées avec TouchStone alors la plateforme de conception permet de définir un protocole expérimental à partir des deux protocoles existants tandis que l'architecture modulaire de la plateforme d'exécution permet d'obtenir un environnement exécutable sans aucun développement logiciel nécessaire. Les résultats de cette expérimentation illustrés par la figure 11 montrent qu'OrthoZoom reste la technique la plus efficace face à toutes ses concurrentes. La réplication et l'extension ont ainsi permis à moindre coût de produire des résultats comparant les quatre techniques sur une même tâche de pointage. Ces résultats sont plus généraux, plus valides et donc plus utiles pour les concepteurs d'interfaces.

5. Conclusion et travaux futurs

Cette approche propose une méthodologie et des outils d'une part pour les chercheurs afin d'améliorer l'accumulation des connaissances et la construction du savoir, d'autre part pour les développeurs afin de faciliter l'intégration de nouvelles techniques dans les interfaces.

D'une part, CIS peut venir assister TouchStone pour le choix des tâches expérimentales valides en explorant différents contextes d'utilisation et en comparant les prédictions avant de réaliser effectivement l'expérimentation. D'autre part, en augmentant le nombre de résultats empiriques et en améliorant leur lisibilité, TouchStone forme un entrepôt de résultats utiles pour l'optimisation des briques de base d'une technique décrite sous forme de graphe CIS.

Les travaux futurs se concentreront sur l'extension de la couverture de CIS. En effet, les prédictions de CIS dépendent de la connaissance que nous avons des lois empiriques sous-jacentes aux actions primitives d'interaction. Par exemple, alors que (Cao *et al.*, 2007) ont proposé les bases d'un modèle permettant d'évaluer le temps nécessaire au tracé d'un geste en fonction de la courbure de ce geste, nous ignorons encore les processus cognitifs impliqués dans le *choix* d'un geste parmi un ensemble de gestes mémorisés. De même, le modèle sur lequel repose le suivi de trajectoire de largeur dynamique exploité dans certaines techniques d'interaction innovantes, comme ControlTree (Appert *et al.*, 2007), reste à étudier.

6. Bibliographie

Accot J., Zhai S., « Beyond Fitts' law : models for trajectory-based HCI tasks », *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press New York, NY, USA, p. 295-302, 1997.

Appert C., Modélisation, Évaluation et Génération de Techniques d'interaction, PhD thesis, Université Paris Sud, 2007.

Appert C., Beaudouin-Lafon M., Mackay W., « Context Matters : Evaluating Interaction Techniques with the CIS Model », *People and Computers XVIII (Proceedings of the 2004 British Computer Society Conference on Human-Computer Interaction.)*, Springer Verlag, p. 279-295, September, 2004.

Appert C., Fekete J., « OrthoZoom scroller : 1D multi-scale navigation », *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, ACM Press, New York, NY, USA, p. 21-30, 2006.

Appert C., Fekete J.-D., « Naviguer dans des grands arbres avec ControlTree », *Proceedings of IHM 2007, 19ème conférence francophone sur l'Interaction Homme-Machine*, ACM, International Conference Proceedings Series, p. 139-142, Novembre, 2007.

Balakrishnan R., « "Beating" Fitts' law : virtual enhancements for pointing facilitation », *International Journal of Human-Computer Studies*, vol. 61, n° 6, p. 857-874, 2004.

Beaudouin-Lafon M., « Instrumental interaction : an interaction model for designing post-WIMP user interfaces », *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press New York, NY, USA, p. 446-453, 2000.

Bier E. A., Stone M. C., Pier K., Buxton W., DeRose T. D., « Toolglass and magic lenses : the see-through interface », *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, p. 73-80, 1993.

Bourgeois F., Guiard Y., Lafon M. B., « Pan-zoom coordination in multi-scale pointing », *CHI '01 : CHI '01 extended abstracts on Human factors in computing systems*, ACM Press, New York, NY, USA, p. 157-158, 2001.

Callahan J., Hopkins D., Weiser M., Shneiderman B., *An empirical comparison of pie vs. linear menus*, ACM Press New York, NY, USA, 1988.

Cao X., Zhai S., « Modeling human performance of pen stroke gestures », *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, New York, NY, USA, p. 1495-1504, 2007.

Card S., Mackinlay J., Robertson G., « A morphological analysis of the design space of input devices », *ACM Trans. Inf. Syst.*, vol. 9, n° 2, p. 99-122, 1991.

Cockburn A., Savage J., « Comparing Speed-Dependent Automatic Zooming with Traditional Scroll, Pan and Zoom Methods », *People and Computers XVII (Proceedings of the 2003 British Computer Society Conference on Human-Computer Interaction.)*, Springer Verlag, p. 87-102, September, 2003.

Fitts P., « The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement », *Journal of Experimental Psychology*, vol. 47, p. 381-391, 1954.

Gamma E., Helm R., Johnson R., Vlissides J., *Design patterns : elements of reusable object-oriented software*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1995.

Gray P., Goodman J., Macleod J., « A Lightweight Experiment Management System for Handheld Computers », *CADUI 2004*, Madeira, Portugal, 2004.

Green T., « Instructions and descriptions : some cognitive aspects of programming and similar activities », *AVI '00 : Proceedings of the working conference on Advanced visual interfaces*, ACM Press, New York, NY, USA, p. 21-28, 2000.

Guiard Y., Beaudouin-Lafon M., Du Y., Appert C., Fekete J., Chapuis O., « Shakespeare's complete works as a benchmark for evaluating multiscale document navigation techniques », *BELIV '06 : Proceedings of the 2006 AVI workshop on BEyond time and errors*, ACM Press, New York, NY, USA, p. 1-6, 2006.

Guiard Y., Beaudouin-Lafon M., Mottet D., « Navigation as multiscale pointing : extending Fitts' model to very high precision tasks », *CHI '99 : Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, New York, NY, USA, p. 450-457, 1999.

Hick W., « On the rate of gain of information », *Quarterly Journal of Experimental Psychology*, vol. 4, n° 1, p. 11-46, 1952.

Hyman R., « Stimulus information as a determinant of reaction time. », *Journal of Experimental Psychology*, vol. 45, n° 3, p. 188-96, 1953.

Igarashi T., Hinckley K., « Speed-dependent automatic zooming for browsing large documents », *UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, ACM Press, New York, NY, USA, p. 139-148, 2000.

John B., Kieras D., « The GOMS family of user interface analysis techniques : comparison and contrast », *ACM Trans. Comput.-Hum. Interact.*, vol. 3, n° 4, p. 320-351, 1996.

Kabbash P., Buxton W., Sellen A., « Two-handed input in a compound task », *Proceedings of the SIGCHI conference on Human factors in computing systems : celebrating interdependence*, ACM Press New York, NY, USA, p. 417-423, 1994.

Kurtenbach G., Buxton W., « The limits of expert performance using hierarchic marking menus », *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press New York, NY, USA, p. 482-487, 1993.

Mackay W., « Which Interaction Technique Works When ? Floating Palettes, Marking Menus and Toolglasses support different task strategies », *Proceedings of the Working Conference on Advanced Visual Interfaces*, p. 203-208, 2002.

Mackay W., Appert C., Beaudouin-Lafon M., Chapuis O., Du Y., Fekete J.-D., Guiard Y., « Touchstone : exploratory design of experiments », *CHI '07 : Proc. ACM Conference on Human factors in computing systems*, p. 1425-1434, 2007.

MacKenzie I. S., Jusoh S., « An Evaluation of Two Input Devices for Remote Pointing », *EHCI '01 : Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, Springer-Verlag, London, UK, p. 235-250, 2001.

Ramos G., Balakrishnan R., « Zliding : fluid zooming and sliding for high precision parameter manipulation », *UIST '05 : Proceedings of the 18th annual ACM symposium on User interface software and technology*, ACM Press, New York, NY, USA, p. 143-152, 2005.

Smith D., Irby C., Kimball R., Harslem E., « The star user interface : an overview », *on AFIPS Conference Proceedings ; vol. 55 1986 National Computer Conference table of contents*, AFIPS Press Arlington, VA, USA, p. 383-396, 1986.

Soukoreff R. W., MacKenzie I. S., « Generalized Fitts' law model builder », *CHI '95 : Conference companion on Human factors in computing systems*, ACM Press, New York, NY, USA, p. 113-114, 1995.

Soukoreff R. W., MacKenzie I. S., « Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI », *Int. J. Hum.-Comput. Stud.*, vol. 61, n° 6, p. 751-789, 2004.

Article reçu le 29 septembre 2008
Accepté le 19 février 2009

Caroline Appert *a reçu le prix Gilles Kahn 2007 pour sa thèse intitulée "Modélisation, Évaluation et Génération de Techniques d'Interaction". Ses travaux s'inscrivent dans le domaine de l'interaction homme-machine et visent à proposer des méthodes d'évaluation et de programmation pour faciliter l'introduction de techniques d'interaction innovantes dans les applications graphiques.*