UNE NOUVELLE ÉTAPE DANS LA CONVIVIALITÉ : LES LOGICIELS AUTO-ÉDUCATIFS

Bertrand IBRAHIM

Résumé :

La convivialité est une notion en évolution permanente, la démocratisation de l'informatique en étant un des moteur essentiels. Il est, de nos jours, commun d'avoir une interface graphique et des aides en-ligne pour la plupart des applications "grand public". Pour être utilisées avec efficacité, ces possibilités impliquent toutefois que l'utilisateur sache plus ou moins ce dont il a besoin et ait déjà une idée assez précise de ce qu'il peut et veut faire. Nous décrivons dans cet article une approche qui vise à enrichir ces applications grand public d'une composante éducative totalement intégrée à l'utilisation des applications en question. Cette approche consiste à intimement combiner utilisation réelle et apprentissage du logiciel, celui-ci prenant l'initiative d'aborder la discussion de certains concepts que l'utilisateur semble ignorer et dont il pourrait tirer profit dans la situation où il est. Nous présenterons ensuite un exemple concret, qui est en cours de développement dans notre groupe de recherche.

1. INTRODUCTION

La convivialité d'une application est, de nos jours, perçue sous trois principaux aspects : la facilité d'utilisation de l'interface homme/machine, la qualité de la documentation accessible en-ligne et la clarté des messages d'erreur fournis par le système en cas de fausse manipulation ou de situations anormales. Principalement depuis que l'avènement de la micro-informatique a permis une démocratisation de l'usage des ordinateurs, les développeurs de logiciels ont en effet réalisé que pour pouvoir vendre un aussi grand nombre que possible de copies de leurs logiciels et ainsi faire un maximum de bénéfices, ils devaient rendre leurs applications facilement utilisables par des personnes n'ayant pas de connaissances très approfondies en informatique.

Il en a résulté des environnements à base de fenêtres, d'icônes et de menus, évitant à l'utilisateur de devoir apprendre un langage de commandes pour pouvoir piloter ces applications. De plus, l'adoption de directives régissant l'apparence et le comportement de l'interface avec l'utilisateur ("look and feel" en terminologie anglo-saxonne) des applications a permis de réduire la courbe d'apprentissage de l'utilisation de nouvelles applications en profitant des connaissances opératoires acquises par l'utilisateur lors de l'utilisation d'autres applications.

A cela s'ajoute généralement la rédaction d'une documentation (guide de l'utilisateur, manuel d'initiation, manuel de référence) et éventuellement le développement d'un tutoriel d'initiation à l'utilisation du produit. Ce tutoriel est parfois fourni (voire vendu) séparément du logiciel, mais il est de plus en plus commun qu'il soit accessible directement depuis l'application même. Cette combinaison n'est toutefois que superficielle, car le tutoriel ne prend pas en considération les activités passées de l'utilisateur.

Une autre approche visant à aider l'utilisateur débutant consiste à intégrer un mécanisme d'aide en-ligne, que l'utilisateur peut invoquer à tout moment, généralement lorsqu'il éprouve des difficultés. Les simples arborescences de pages d'explications, ou parfois des hypertextes, à travers lesquels l'utilisateur peut naviguer, commencent maintenant à être remplacés par des aides contextuelles qui prennent en compte les dernières actions effectuées. L'utilisateur peut ainsi obtenir des explications complémentaires sur l'erreur qu'il a commise ainsi que des suggestions de "remédiations" [1][5][9]. Encore faut-il, pour que cette information soit pertinente, que l'utilisateur ait utilisé la bonne commande. Un tel mécanisme ne sera, en effet, pas d'une très grande utilité si l'utilisateur ne sait pas ce dont il a besoin ou ne connaît pas le concept qui pourrait lui être utile dans la situation où il se trouve.

Il ne faut pas oublier qu'il y a presque toujours un modèle sous-jacent à toute application informatique. Ce modèle était généralement déjà présent à l'esprit des développeurs lorsqu'ils ont commencé le développement de l'application et la compréhension de ce modèle, qui est rarement explicite, facilitera grandement l'utilisation de l'application.

Par exemple, pour une application de courrier électronique, le modèle sous-jacent inclus les notions de pseudonyme, de classeur pour l'archivage, d'adresses électroniques (locales ou à distance), de recherche sélective dans les archives, de composition de messages, de réponse à des messages reçus, de retransmission d'un message reçu à une autre per

sonne, etc. Ces notions sont liées entre elles : peut-on répondre à un message archivé ? Peut-on utiliser une adresse électronique présente dans un message pour définir un pseudonyme ?

Pour qu'une aide à l'utilisateur soit efficace, il est bon d'aller au-delà de la simple explication des commandes opératoires du système. L'enseignement, par exemple par le biais d'un tutoriel, du modèle sur lequel est basée l'application devrait être à même de considérablement augmenter/améliorer la compréhension que l'utilisateur a du système qu'il utilise.

Il est toutefois rare qu'un utilisateur ait la patience de s'installer face à un tutoriel pour apprendre tous les aspects d'un logiciel, aspects dont il ne perçoit pas toujours l'utilité. De plus, certains concepts "avancés" ne seront probablement pas d'une grande utilité pour un débutant, mais ledit débutant n'est pas forcément capable de s'en rendre compte par lui-même. Il n'est donc pas aisé de réduire la charge cognitive de l'apprentissage de l'application à son strict minimum lors de l'utilisation d'un tutoriel indépendamment du logiciel dont il est sensé enseigner l'usage.

2. LOGICIELS AUTO-ÉDUCATIFS

C'est pour sublimer ces contraintes que notre groupe de recherche s'est intéressé, en collaboration avec l'équipe du professeur Alfred Bork, de l'université de Californie à Irvine, au concept de logiciel que nous avons baptisé d'auto-éducatif (en anglais, "self-instructional software"). Ce concept vise le développement de logiciels d'utilité générale dans lesquels il n'y aurait pas de séparation entre les fonctions d'apprentissage et les fonctions purement opératoires, toute situation d'utilisation devenant alors une situation potentielle d'apprentissage. Ceci a été décrit, par certains auteurs [10], comme des environnements "sans rupture" (en anglais, "seamless" environments), c'est-à-dire des environnements de travail dans lesquels il est difficile de distinguer entre les moments où l'on travaille et les moments où l'on apprend.

Appliquée aux logiciels, cette approche consiste à se préoccuper, tout au long du développement d'un logiciel, des différentes catégories d'utilisateurs et d'inclure dans le logiciel tout ce qui est nécessaire pour que les utilisateurs puissent, quels que soient leur bagage et leurs connaissances préalables, utiliser et, le cas échéant, apprendre à utiliser efficacement ce logiciel et les concepts qui le sous-tendent. Le logiciel prend

alors un rôle actif : l'utilisateur garde le contrôle du déroulement du programme, mais le logiciel peut prendre des initiatives et faire des suggestions que l'utilisateur peut choisir d'ignorer ou de suivre. L'ordinateur n'est plus maître ou esclave ; il devient un collaborateur.

Cette approche a un intérêt pédagogique indéniable puisqu'elle permet de fournir des explications ou d'introduire de nouveaux concepts "in situ", c'est-à-dire dans une situation où ils ont un sens, au moment où ils sont le plus utiles et, par conséquent, le plus à même d'être compris et assimilés. Elle permet aussi de proposer à l'utilisateur d'essayer une simulation avant d'utiliser pour la première fois une primitive dont le mauvais usage pourrait mettre l'utilisateur dans l'embarras. Cela nécessite de collecter, pour chaque utilisateur, des informations tout au long de l'utilisation du produit afin de déterminer le "profil" de cet utilisateur et d'adapter en conséquence le dialogue homme-machine. Le facteur temps constitue un aspect à prendre aussi en considération car, même lors d'une utilisation fréquente, l'usage de certaines fonctions rarement utilisées peut comporter des difficultés. On peut ainsi imaginer que les savoirs acquis par un utilisateur se périment s'ils ne sont pas mis en œuvre pendant un certain laps de temps.

Cette notion de profil de l'utilisateur permet aussi de concevoir une interface adaptative qui évolue en fonction de l'évolution de l'utilisateur, au fur et à mesure que celui-ci acquiert de l'expérience dans l'utilisation du logiciel. On peut, par exemple, envisager d'afficher des menus très explicites pour un utilisateur débutant, puis rendre ces menus de plus en plus concis au fur et à mesure que l'utilisateur se sera habitué à les utiliser. Nous manquons toutefois d'expérience pratique sur ce plan et la question reste ouverte de savoir si une telle interface "évolutive" constitue réellement une amélioration. Il conviendra, en effet, de vérifier si cette technique n'a pas un effet psychologique déstabilisant allant à l'encontre du but visé. Cette notion d'interface adaptative n'est pas nécessairement liée au concept de logiciel auto-éducatif, mais l'infrastructure nécessaire pour développer ces derniers permet de facilement mettre en place cette adaptativité.

Notre approche va quelque peu à l'encontre de la tendance actuelle en matière de développement d'interfaces utilisateur qui est reflétée par l'utilisation de plus en plus répandue de générateurs d'interfaces graphiques. La tendance générale est, en effet, de clairement dissocier la spécification et l'implantation de cette interface de la spécification et de l'implantation des fonctions opératoires que cette interface doit activer. Cette

dichotomie franche a pour intérêt de faciliter le développement de l'application ainsi que la standardisation de l'interface de toute une série d'applications partageant alors la même apparence ("look and feel") et permettant ainsi à l'utilisateur de réutiliser, pour de nouvelles applications, certaines connaissances opératoires qu'il aurait acquises précédemment lors de l'utilisation d'autres applications.

A l'opposé, notre approche repose sur une étroite dépendance entre la spécification de l'interface utilisateur et celle les fonctions opératoires et des concepts de l'application. Nous devons donc, pour cela, utiliser une méthodologie de développement différente qui soit mieux appropriée à ce contexte, les méthodologies et outils classiques de développement d'interfaces n'étant pas adaptés aux considérations d'ordre pédagogique.

3. LE PROJET DE MESSAGERIE AUTO-ÉDUCATIVE

Afin de mettre en œuvre ce concept de logiciel auto-éducatif, nous avons choisi de développer un système de courrier électronique [2][3] car c'est un outil qui est, pour l'instant, principalement utilisé par des informaticiens mais qui est promis à une large diffusion, bien au-delà des cercles de professionnels de l'informatique. Il est donc important qu'un tel outil puisse être à la portée d'à peu près tout le monde.

Tout utilisateur de courrier électronique, qu'il soit novice ou professionnel, voudra commencer à recevoir et à envoyer des messages aussi rapidement que possible. Bien d'autres fonctions sont disponibles, mais elles sont secondaires en regard de ces deux premières. La question par laquelle nous avons commencé consistait donc à déterminer de quelle façon il était possible de combiner l'apprentissage de l'envoi et de la réception de messages avec l'utilisation de ces mêmes fonctions. Dans notre cas, par exemple, le système prend l'initiative de déposer un message de bienvenu dans la boite aux lettres vide d'un utilisateur qui n'aurait pas encore de correspondant. D'une manière similaire, le système est à même de proposer l'adresse d'un correspondant local qui se serait porté volontaire pour aider ou conseiller des débutants. Le système est aussi capable de proposer l'adresse d'un correspondant fictif auquel l'utilisateur pourrait envoyer un message pour s'exercer.

D'une façon générale, nous avons essayé de concevoir l'interface-utilisateur de sorte qu'un concept ne soit abordé que lorsque le besoin s'en faisait sentir. Par exemple, la notion de classement des messages dans des "classeurs" ("folders" en anglais) n'est abordée par le logiciel

qu'à partir du moment où l'utilisateur a suffisamment de messages pour qu'un classement se justifie. Il va de soi que l'utilisateur peut invoquer la fonction de classement à tout moment, même si le logiciel ne l'a pas encore considéré comme nécessaire. Tout cela fait partie de notre conception du logiciel comme un assistant "intelligent", c'est-à-dire obéissant, mais pas nécessairement passif pour autant.

Comme cela a été mentionné précédemment, le logiciel doit maintenir un "profil" de l'utilisateur afin de pouvoir déterminer s'il y a lieu d'activer certaines parties didactiques du programme. Dans le cas de la messagerie, les informations conservées dans ce profil comprennent des éléments tels que :

> un message a été lu, un message a été envoyé, connaît une adresse locale, connaît une adresse à distance, nombre de fois que cet utilisateur a utilisé le système, liste de commandes d'édition utilisées, a sélectionné des messages dans un classeur, nombre de fois qu'un message d'origine a été inclus dans une réponse, liste des commandes utilisées avec succès, etc.

Un prototype de ce système est en cours d'achèvement. Sa spécification ainsi que sa réalisation ont été faites à l'aide d'un environnement de développement de didacticiels faisant aussi l'objet de nos activités de recherche [6][7][8][4]. Nous développons en parallèle deux versions : l'une pour station de travail Sun utilisant le système de messagerie EAN et l'autre pour Vax/VMS utilisant Vax-Mail. La réalisation en parallèle de deux implantations nous permet d'assurer un haut niveau de portabilité de l'application ainsi produite.

Bertrand IBRAHIM

Département d'Informatique
Université de Genève
24, rue du Général Dufour
1211 Genève 4, SUISSE

4. RÉFÉRENCES

[1] Axion et al. ; "EuroHelp : Developing Intelligent Help Systems" ; rapport sur le projet P280 ESPRIT EUROHELP ; Joost Breuker, éditeur, 1990 EC, Copenhague, Amsterdam, Manchester, Leeds.

[2] ALFRED BORK, BERTRAND IBRAHIM, BIRGIT LAUSTSEN, BERNARD LEVRAT ; "A Self-Instructional Mailer" ; WCCE/90, Résumés de Conférence, North-Holland, 1990

[3] ALFRED BORK, BERTRAND IBRAHIM, BIRGIT LAUSTSEN, BERNARD LEVRAT ; "A Self-Instructional Mailer" ; Ninth International Conference on Technology and Education, Conference Proceedings, Paris, 16-20 mars 1992, pp. 871-873.

[4] ALFRED BORK, BERTRAND IBRAHIM, BERNARD LEVRAT, ALASTAIR MILNE, RIKA YOSHII ; "The Irvine-Geneva Course Development System" ; IFIP Congress '92, 7-11 septembre 1992, Madrid, Espagne, Vol II, pp. 253-261.

[5] W. DZIDA, S. HERDA, W.D. ITZFELDT ; "User perceived quality of interactive systems" ; IEEE Transactions on Software Engineering, Vol 4, No 4, 1978, pp. 270-276.

[6] BERTRAND IBRAHIM ; "Software engineering techniques for CAL" ; Education & Computing, Vol 5, pp. 215-222, Elsevier Science Publishers, 1989.

[7] BERTRAND IBRAHIM, ALAIN AUBORD, BIRGIT LAUSTSEN, MICHAEL TEPPER ; "Techniques de Génie Logiciel pour l'EAO" ; Conférence sur "Enseignement et Apprentissage avec l'Ordinateur", Martigny, 23-24 novembre 1989, pp. 120-129.

[8] BERTRAND IBRAHIM, ALAIN AUBORD, BIRGIT LAUSTSEN, MICHAEL TEPPER ; "Courseware CAD" ; WCCE/90, Sydney, 9-13 Juillet 1990, Conference Proceedings, pp. 383-389, North-Holland, 1990.

[9] M. PRAGER, D.M. LAMBERTI, D.L. GARDNER, S.R. BALZAC ; "REASON : an Intelligent User Assistant for Interactive Environments" ; IBM Systems Journal, Vol 29, No 1, 1990

[10] MARY S. TRAINOR ; ADCIS News, Vol 24, janvier 1991.