Formation, Audit, Conseil, Développement, UX

SILVERLIGHT – WPF – C#

Articles et Livres Blancs gratuits à télécharger www.e-naxos.com
Dot.Blog, le blog www.e-naxos.com/blog

MICROSOFT MVP Silverlight 2011, MVP Client App Dev 2010, MVP C# 2009

LE DESIGN DES APPLICATIONS

[Concevoir des interfaces efficaces et vendeuses]

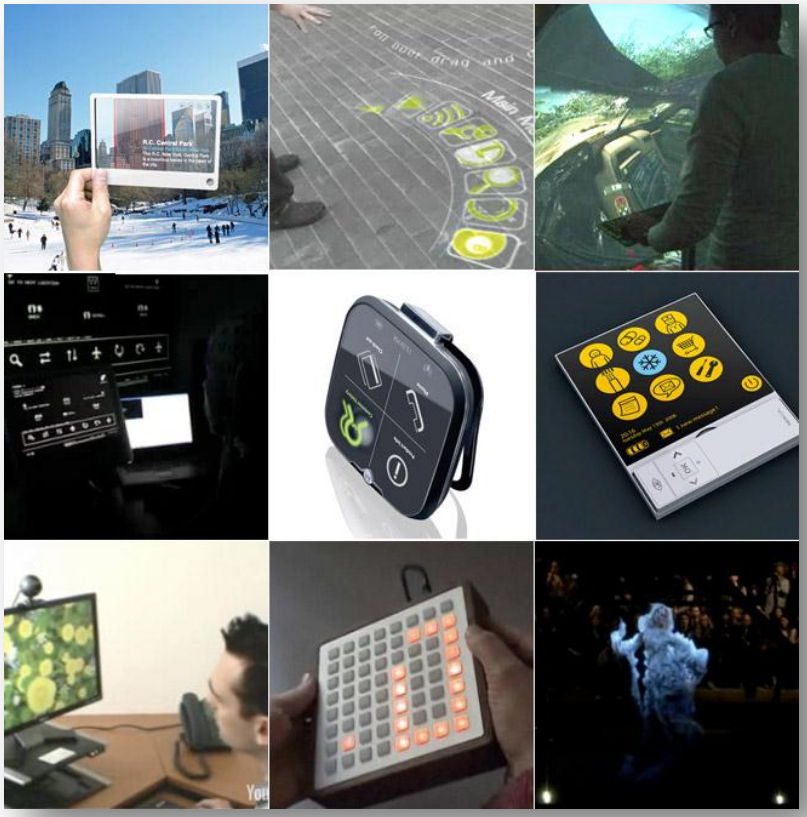
Version 1.0 Septembre 2011

Sommaire

Préambule

Parler de code c'est bien, parler de patterns ou des nouveautés de Silverlight c'est encore mieux, mais parler de Design c'est aujourd'hui tout aussi essentiel. C'est pourquoi j'aborde le sujet régulièrement… Dans le présent article je vous propose de faire le point sur la création des interfaces modernes, l'importance de cette tâche, les motivations de ce mouvement inéluctable.

Il ne s'agit pas d'un « cours » sur le Design qui réclamerait d'entrer plus en profondeur sur chaque point abordé ici, mais d'une présentation des grands concepts qui vous aideront à mieux concevoir vos UI et à garantir un UX de qualité à vos utilisateurs.



La fin d'un mythe, le début d'une aventure

La première chose à comprendre et à admettre c'est que notre beau métier qu'est l'informatique a eu tendance ces dernières décennies à laisser croire à certains qu'ils étaient un peu les "maîtres du monde". C'est un peu vrai… Sans nous, sans notre code, le Monde ne serait pas aujourd'hui tel qu'on le connaît : pas d'Airbus ni d'avion de chasse in-pilotable sans ordinateur (donc sans logiciel), pas d'Internet ni de Web 1 ou 2, pas d'IPhone, ni d'IPad, pas d'information en temps réel, pas de Webcams aux quatre coins du monde, rien, pas même de distributeur de billets, pas de CD, ni de DVD, pas de GPS. Rien. Enfin si, le monde tel qu'il était à la fin des années 50.

Bref dans ce monde de l'information, l'informaticien était un petit roi. Les électroniciens vous diront que ce sont eux les vrais maîtres du monde, sans circuits intégrés, notre métier n'existerait même pas (cogiter toute la journée sur la beauté algorithmique de programmes virtuels sur une machine de Turing purement idéale n'aurait pas changé le Monde).

La vraie **cassure conceptuelle**[1], le vrai nouveau paradigme "à avaler" c'est que depuis quelques années l'informaticien est devenu un OS des temps modernes (sans jeu de mot, ou presque).

Mais il y a encore plus grave : la chute du petit roi ne s'arrête pas là, les vrais maîtres du monde aujourd'hui en matière de technologie ce sont les Designers !

L'Iphone offre-t-il une meilleure écoute que la concurrence ? Non. Offre-t-il une meilleure couverture ? Non, cela ne dépend pas de lui.

Offre-t-il des fonctions téléphoniques plus évoluées que la concurrence ? Non plus, passer un coup de fil se fait aujourd'hui comme dans les années 60 : on décroche et on compose son numéro, ça sonne, on attend que l'autre réponde…

Les problèmes d'antenne de modèles récents, tout comme le fait qu'on reconnait un utilisateur iPhone car c'est le seul à se trimbaler avec son chargeur et à chercher une prise électrique dès qu'il arrive chez vous, démontrent même que comparé à la concurrence comme Nokia les Smartphones Apple sont nettement de qualité moindre. Ils sont d'ailleurs fabriqués en Chine, comme les concurrents, par des entreprises qui ne sont pas plus respectueuses des droits de l'homme, du travail des enfants et de la protection de l'environnement que celles qui fabriquent pour Nokia, Sony-Ericsson ou d'autres…

Pourtant ce sont là des points vitaux pour un téléphone ! Non, réaffirmons-le, *sur l'essentiel de ce qui fait un téléphone moderne l'IPhone ne fait rien de tout cela mieux que les autres* (c'est même plutôt l'inverse !).

Mais il a un Design.

Et ce n'est qu'un exemple parmi des milliers.

[1] La « Cassure Conceptuelle » est un concept que j'ai introduit dans le billet « Le défi des nouvelles interfaces Silverlight et WPF - La cassure conceptuelle » du 1er septembre 2009.

On pourrait parler des voitures qui se vendent mieux que les autres, et de milles objets, de la cafetière aux couverts de table, qui eux aussi sont tombés sous la domination du Design. Ce qui a fait le succès d'un Ikea sur ces concurrents "low cost" ? … Le Design.

(Ci-contre un presse-agrume signé Philippe Starck)

De Charybde en Sylla, notre pauvre informaticien ne cesse de dégringoler de son trône !

Alors certains changements se font dans la douleur au lieu de se faire dans la joie. Il y a de la résistance, du passéisme, on devient réactionnaire, méfiant, traitant tout changement de "mode" pour en minimiser l'impact et l'importance, vénérant des dieux déchus, des langages oubliés. En un mot on se ringardise.

> ***« Puisque ces mystères me dépassent, feignons d'en être l'organisateur »***

Heureusement il existe une issue plus heureuse ! N'oublions pas cette pensée sublime de Cocteau.

Le même disait aussi

> ***« L'avenir n'appartient à personne. Il n'y a pas de précurseurs, il n'existe que des retardataires. »***

Ce qui, quand on prend la mesure est à la fois d'une profondeur inouïe et un éclairage optimiste sur notre fonctionnement : il nous suffit de vivre avec notre temps pour ne pas être ringard et pour passer pour des visionnaires... Mais nous sommes tellement englués dans le passé que même ceux qui croient vivre "au temps présent" se trompent : le présent n'existe pas, puisque dès qu'on l'évoque il s'agit déjà du passé.

Le poète est souvent un sage qu'on gagne à écouter…

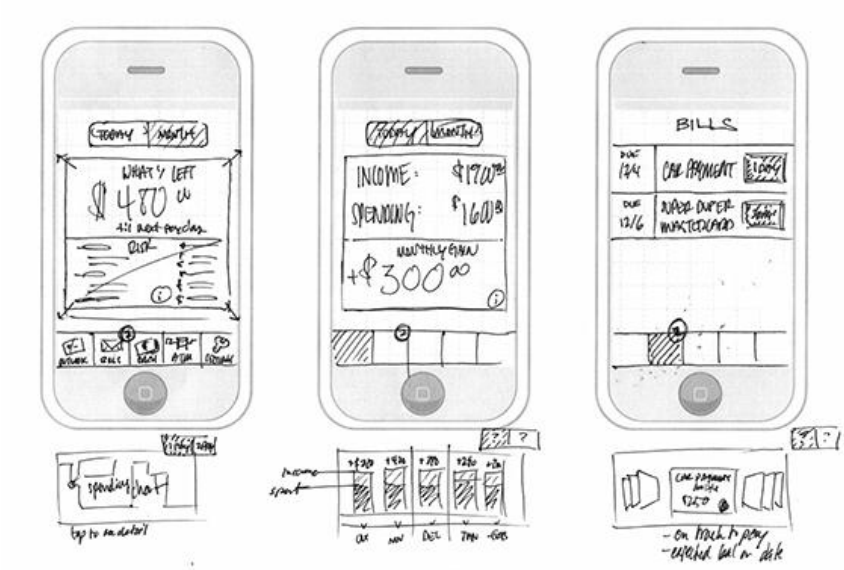
Le rapport avec le sujet ? C'est que les Designers ne peuvent pas faire ce qu'ils veulent, ils ne sont rien sans le fonctionnel ! C'est d'ailleurs ce qui fait d'un simple créateur ou dessinateur un vrai « Designer » *c'est la prise en compte des contraintes d'utilisation* !

Si tout le monde sait ce qu'est une fourchette ou un presse-agrume, et s'il est donc facile à un Starck d'imaginer de nouveaux designs pour de tels objets, en matière d'informatique les choses sont plus complexes. C'est pourquoi les vrais Designers d'interface sont encore bien rares.

Les informaticiens peuvent reprendre le dessus s'ils ouvrent les yeux, s'ils sortent de leur grotte et tombent la carapace.

Un IPhone dont le contenu ne serait que fakes et qui ne passerait aucun coup de fil pour de vrai, où les jeux ne seraient que des plans fixes comme des affiches alléchantes mais sans le film qui va avec et où les applications ne seraient que des dessins d'interface sous Photoshop, vous croyez que ça se vendrait ?

Non. Bien entendu. Et pourtant dans un monde de Designers sans informaticiens, l'IPhone ressemblerait à cette description et serait totalement inutile…



Le vrai pouvoir est au fonctionnel, encore faut-il ne pas se laisser déposséder de ce pouvoir.

Et si les informaticiens d'aujourd'hui ne sont pas les créateurs de cette grande tendance du Design, ils peuvent largement en être les organisateurs (Cf. La citation de Cocteau plus haut…) ! La maîtrise du fonctionnel, la connaissance du besoin de l'utilisateur, aucun infographiste ne l'a. Aussi talentueux soit-il.

Certes il semble aussi difficile de faire apprendre l'art noble du Design à un informaticien que d'expliquer les merveilles de C# à un infographiste. Je vous l'accorde. Mais il n'est pas interdit d'avoir de l'imagination, et, avec les bons outils et les bons collaborateurs l'informaticien du 3ème millénaire

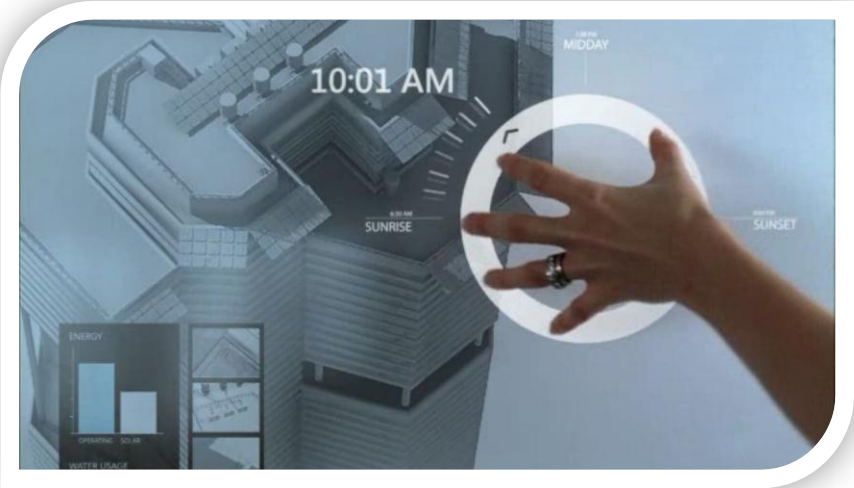
peut fort bien devenir, enfin, le maître du monde qu'il pensait être, ou tout du moins *s'associer intelligemment avec les Designers pour accéder ensemble au trône*…

Mais il reste à comprendre l'essentiel, le fait que le Design n'est pas un accessoire, pas plus qu'une mode, *c'est un acte essentiel de la conception d'un produit*, qu'il s'agisse d'une voiture, d'une fourchette ou d'un logiciel. **Les Designers ne sont pas des putschistes venant nous voler la vedette, ce sont des alliés inespérés pour nous aider à conquérir de nouveaux utilisateurs**.

Pour nous tout commence donc avec les interfaces graphiques…

L'importance des interfaces utilisateur

Pourquoi sont-elles si importantes ?

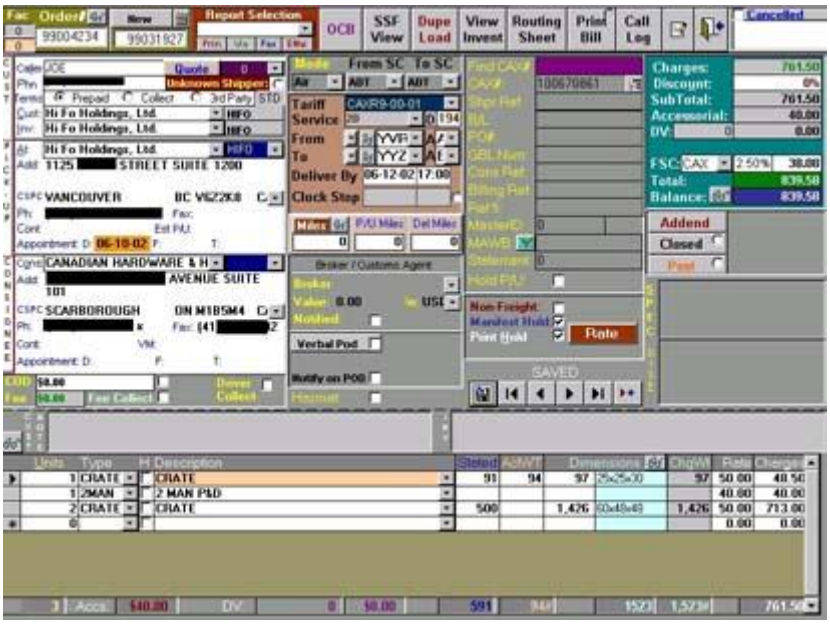


Parce que **c'est la seule partie d'un programme que l'utilisateur peut voir et toucher** !

Parce que nous connaissons tous des applications qui possèdent des fonctionnalités incroyables mais qui ne connaîtront jamais le succès à cause de leur mauvaise interface…

Parce que l'informaticien a toujours du mal à intégrer que l'utilisateur va peut-être passer 8 heures par jour à se fatiguer sur une interface qu'il n'aime pas…

Les utilisateurs ne sont que des humains comme les autres et ils jugent souvent une application en quelques minutes (souvent moins!), juste sur une "première impression". Ce premier contact aura un impact direct et immédiat sur la diffusion de l'application (sa vente, sa fréquentation si c'est un site Web…). Pire, tout cela se fonde presque uniquement sur l'interface utilisateur !



Aimeriez-vous passer des journées entières sur le logiciel ci-dessus ?

Nous sommes plongés, de gré ou de force, dans un monde d'images manipulées, un monde d'apparence où il devient presque impossible de différencier une image réelle d'une image fabriquée. En revanche, et à plus forte raison, habitués que nous sommes à des images "léchées", aux filles aux jambes improbables *photoshopées* des magazines, *il nous est très facile de voir au premier coup d'œil un logiciel moche au look ringard* !

Je connais la chanson pour l'avoir entendu milles fois : "les logiciels hyper lookés sont souvent vides fonctionnellement, moi j'ai fait un soft qui a la super fonction Z que personne n'intègre !". C'est peut-être vrai, mais n'empêche votre zinzin ne se vendra pas. Le relooking d'une application n'est pas une option, c'est une évidence, une partie non négociable de sa fabrication. Le reste n'est qu'excuse pour tenter de rester vainement le maître d'un monde qui n'existe plus…

L'utilisateur potentiel doit aussi percevoir une certaine familiarité, doit se sentir compris, et cela,

l'interface peut le communiquer en quelques secondes mieux que 200 pages d'une documentation qu'il ne lira pas et qui tentera de lui faire comprendre la beauté de la fonction Z que personne d'autre n'implémente.

Il faut aussi prendre conscience que la plupart des utilisateurs compareront votre application à d'autres.

Souvent l'informaticien, très (trop ?) content de son "œuvre" oublie de regarder ce qui se fait ailleurs et comment cela est fait. Croyez-vous que Nokia lance la fabrication d'un nouveau téléphone sans avoir démonté tous ceux de ses concurrents avant ? Qui eux-mêmes en ont fait autant.

D'ailleurs vous. Oui vous qui lisez ces lignes. Le dernier logiciel que vous avez créé, avez-vous ne serait-ce qu'une fois tenté de voir comment des produits similaires ou concurrents répondaient au même besoin ? Quel était celui qui se vendait le mieux ? Pourquoi ? Je parie qu'une majorité de lecteur ne pourra que répondre « non ». Hélas…

L'informaticien n'est pas dans une logique industrielle car l'industrie est un système de production de masse alors que l'informatique réinvente tout pour chaque logiciel, c'est de l'artisanat. On entend parfois que c'est une passion de geeks, de no-live, un "trip" ultra-perso. Mais c'est une vision dépassée de notre profession. *Passer de cet isolement à un regard ouvert sur ce qui se fait ailleurs est un bon commencement pour améliorer l'interface de ses applications et devenir un informaticien qui vit avec son temps.*

Qu'est ce qui fait une bonne interface ?

S'il y avait une recette pour fabriquer des IPhones, tout le monde en vendrait sous toutes les marques. Hélas les "bonnes" interfaces sont comme les musiques qui rencontreront le succès, s'il y avait une "recette" cela se saurait depuis longtemps !

Mais en revanche si on ne sait pas dire exactement ce qu'il faut faire pour créer une bonne interface, on peut en cerner les principales qualités :

- Une bonne interface semble **facile**, l'application apparait **intuitive** sans lire de manuel;
- Une bonne interface est **vendeuse**, au premier contact. Mais elle se révèle aussi **productive** au fil du temps pour les utilisateurs avancés;
- Une bonne interface permet à l'utilisateur **d'accomplir les tâches de la façon qu'il souhaite les accomplir** et non en étant forcé de suivre une logique rigide imposée par le concepteur;
- Une bonne interface est **cohérente**. La cohérence doit s'étendre à toute l'application mais aussi à tout l'environnement de l'utilisateur et au cadre socio-culturel d'utilisation du logiciel, au profil psychologique des utilisateurs potentiels.

Quelques concepts importants de Design

Connaître ses utilisateurs

Cela peut sembler idiot, mais le plus souvent l'informaticien, même talentueux et ayant une bonne idée de logiciel, n'a jamais vu ni rencontré les utilisateurs potentiels ! Je ne parle pas des informaticiens créant des logiciels dits "standards" comme des comptabilités chez de gros éditeurs et qui travaillent presque à la chaîne, dans un total isolement par rapport à l'utilisateur final.

Bien connaître l'utilisateur final d'une application permet de définir les buts du Design de l'interface et peut modifier les lignes et la stratégie du développement lui-même.

Une application peut se destiner à des utilisateurs ayant un niveau technique et une habileté totalement différents des vôtres. Il faut en tenir compte et cela ne peut se faire qu'en connaissant parfaitement la cible concernée. D'où la nécessité souvent de définir des profils de personnages fictifs et de se mettre dans « leur peau » pour tester l'ergonomie et la conception de l'interface d'un logiciel (Robert le mécanicien pas geek du tout qui doit gérer sa comptabilité, Mauricette, sa secrétaire, la cinquantaine, qui a commencé sur une machine à écrire électrique, ou Carole, sa fille qui lui donne un coup de main de temps en temps, la vingtaine, écouteurs d'Ipod collés aux oreilles, saisissant des factures un œil sur son BlackBerry et les messages qui arrivent…). Ces profils s'appellent des Personas[2].

Une application peut être conçue pour des utilisateurs ayant un mode de pensée totalement différent du vôtre. *Ne pas savoir comment l'utilisateur "fonctionne", comment il pense, ses réflexes, ce qu'il aime, ce qu'il déteste, c'est passer à côté de l'essentiel et forcément prendre le risque de ne pas le séduire ni le satisfaire* !

Les buts de l'utilisateur

Un utilisateur n'est jamais devant une application pour le "fun". Ou alors c'est un informaticien et c'est une autre histoire… Un utilisateur se sert d'un logiciel pour **accomplir une ou plusieurs tâches précises**. Si c'est un jeu, ce n'est pas non plus pour le "fun" qu'il s'en sert, mais bien égoïstement pour se procurer du plaisir. Il n'y a pas d'acte gratuit chez l'humain.

Connaître parfaitement ces tâches que l'utilisateur cherchera à accomplir permet de proposer des méthodes de travail en accord avec ses buts, sa façon de procéder, de penser. Le fameux sentiment de familiarité, de proximité que j'évoquais plus haut peut naître de cette connaissance et de l'adéquation du Design avec cette dernière.

Un utilisateur n'aime pas se sentir "idiot". Une bonne interface sait le guider s'il ne sait pas comment faire. Elle évite aussi les messages brimant du genre "Action interdite!" ou "Saisie non valide!". Ce sont pourtant des choses qu'on voit tous les jours !

L'utilisateur est parfois forcé d'utiliser un logiciel, auquel cas plus ce dernier saura le séduire, mieux cela sera. Mais souvent l'utilisateur choisit un logiciel parce qu'il pense que cela va l'aider à faire quelque chose qu'il ferait mal ou moins bien sans l'aide d'un ordinateur et d'une application étudiée. *Il ne faut pas le décevoir*…

[2] Les personas sont des représentations fictives mais concrètes des utilisateurs pour lesquels le produit est conçu. Ils fournissent aux développeurs une référence pour définir les fonctionnalités et les scénarios d'utilisation.

Les deux types d'utilisateurs

Il y a *l'autonome* et *le survivant*.

L'autonome prend sur lui les fautes et ne les rejette pas sur le système (téléphone, PC, logiciel…). Il sait en apprécier les petits avantages et a tendance à minimiser les gros défauts. Il se débrouille. C'est le genre d'utilisateur qui, si vous lui montrez les faiblesses du système qu'il utilise vous dira "oui mais regarde, quand j'appuie là ça fait clignoter le clavier !". Les informaticiens sont comme ça. C'est aussi le profil des "power users".

Le survivant voit le monde autrement : lui il sait qu'il y a des problèmes dans le système mais ne sait pas comment les régler. Il aimerait d'ailleurs posséder un système *plus simple*, *moins bogué*, plus *proche de ses besoins*, mais il fait avec ce qui existe (ou avec ce qu'on lui a mis entre les mains). Quant au petit gadget qui fait clignoter le clavier, "même mon ours en peluche quand j'avais 4 ans savait faire mieux!".

La grande majorité des utilisateurs appartiennent à cette seconde catégorie !

L'habileté

Les utilisateurs sont des débutants pendant une période qui est généralement très courte. Passé un certain cap, ils se contenteront des solutions qu'ils ont trouvées pour accomplir les tâches usuelles et *n'iront pas chercher plus loin*.

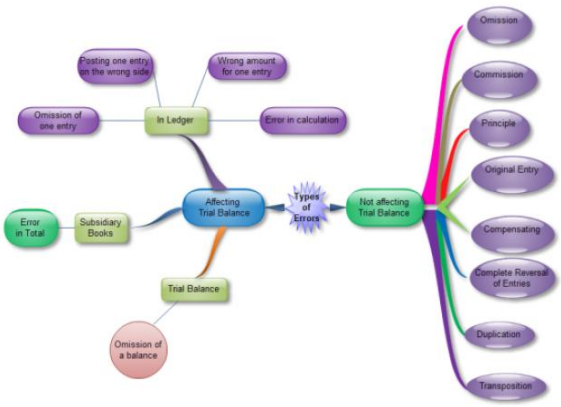
Seule une petite fraction minoritaire des utilisateurs deviendra vraiment "expert". Le genre d'utilisateur qui peut vous faire voir des raccourcis ou des astuces d'utilisation que vous-mêmes en tant que concepteur du logiciel n'avez jamais imaginé !

En fait, la majorité des utilisateurs seront dans la moyenne, c'est la courbe de Gauss qui s'impose avec son corollaire : une majorité dans le centre de la cloche, une minorité aux extrémités.

Quelques cas font exceptions à cette règle : les applications publiques, de type annuaire téléphonique ou distributeur de billets, et les applications que les utilisateurs n'aiment pas. Dans ces deux cas la majorité des utilisateurs ne fera aucun effort pour comprendre le système et rejettera systématiquement la faute sur ce dernier.

Les schémas directeurs

Un utilisateur va approcher une application avec un certain nombre d'*a priori*, de réflexes, de **schémas directeurs** qui dépendent de ses *connaissances passées relatives au domaine du problème traité par le logiciel*.



Se fondant sur ces schémas, *l'utilisateur va s'attendre, inconsciemment la plupart du temps, à ce que l'application suive un certain "modèle", un certain "schéma de fonctionnement"*.

Pensez à un comptable, lorsqu'on est passé des livres et journaux physiques à l'informatique il y a eu des grincements de dents… Un informaticien est parfaitement capable de créer une comptabilité qui réclame des connaissances mathématiques du niveau 6ème (additions, soustractions, divisions et multiplications). Mais en revanche il n'est pas comptable et ne connaît pas les habitudes des gens de cette profession. Les premiers logiciels ont reçu un accueil très mitigé.

Pensez aussi à un médecin. Habitué à une relation personnelle avec son patient et à qui "le progrès" impose un "tiers" s'installant entre lui et le patient : l'ordinateur et le logiciel "médical". Ayant fait partie des pionniers de cette branche particulière des logiciels verticaux dans les années 80 je peux vous garantir que les prévisions et les *a priori* de tous les informaticiens ont été balayés très vite lorsque leurs logiciels ont été confrontés aux utilisateurs !

Comment une consultation se déroule-t-elle, dans quel ordre un médecin traite-t-il le dossier du patient, combien de temps acceptera-t-il de passer à frapper sur le clavier alors que le patient se tient là, assis en face de lui ou allongé sur la table d'auscultation ? Comment va-t-il faire en visite ?

Les dossiers des patients en papier se transportaient facilement dans la mallette, on y écrivait deux lignes au chevet du patient et quand on remettait le dossier dans l'armoire en rentrant il était à jour. Si le patient téléphonait la secrétaire sortait son dossier de l'armoire pour lui répondre ou le transférer au médecin, pas besoin de "synchronisation", de "réseau", de "portable", ni de Wi-Fi et encore moins d'Internet… Quant aux "backups", la photocopieuse utilisée par la secrétaire était parfaite, pas besoin de disques externes, de NAS, de serveurs, de "scheduler" pour "programmer" les sauvegardes, etc… Et même en cas d'incendie, les piles de papier résistent bien mieux que les ordinateurs (expérience douloureuse faite avec l'incendie d'un appartement que j'occupais il y a de cela une quinzaine d'années).

L'informatique médicale était au final une gêne plus qu'une amélioration !

Cela reste vrai aujourd'hui, et fonctionnellement les logiciels modernes se sont adaptés en offrant nettement moins de possibilités. Il s'agit pour la majorité de "gros bloc-notes" flanqués d'une version électronique du Vidal (le dictionnaire des médicaments) et surtout accompagnés d'un module gérant la carte Vitale (seul vrai argument ayant poussé les médecins à s'informatiser, car si la science pure les enquiquine, les sous, ça les intéresse !). D'une application "scientifique" cherchant à apporter un soutien "intelligent", voire une aide au diagnostic et des systèmes experts, nous en sommes arrivés à des fourre-tout électroniques gérant la relation comptable avec la Sécu. Le médecin n'était pas le "scientifique" qu'on imaginait, il n'avait pas le temps qu'on croyait pour s'occuper d'un logiciel en même temps que d'un patient, et pour lui se faire payer était plus important que de pondre des statistiques sur les cas de grippes qu'il avait vu dans l'année. Enorme désillusion.

Surtout : **mauvaise approche de l'utilisateur** !

Erreur de casting, Le « profilage » de l'utilisateur avait été mal fait au départ.

Les premiers logiciels médicaux ont été développés avec l'aide de médecins "geeks" bricolant souvent eux-mêmes des petits logiciels, *les informaticiens croyaient voir à travers eux tous les médecins, ils ne voyaient que des exceptions*, pas la règle. L'erreur a souvent été fatale pour beaucoup de sociétés[3] !

Ainsi, *connaitre les schémas directeurs utilisés consciemment et inconsciemment par l'utilisateur n'est pas un luxe, c'est une obligation*. Ne pas se tromper de modèle et choisir un panel d'utilisateurs représentatifs du milieu de la cloche de la courbe de Gauss est tout aussi important… Créer des profils types répondant à la réalité est une tâche essentielle lors du Design.

Cognitive Friction

Ce terme provient du second livre de Alan Cooper, le père de Visual Basic, "The inmates are running the asylum: why High-Tech products drive us crazy and how to restore the Sanity" - *Les aliénés dirigent l'asile : Pourquoi les produits High-Tech nous rendent fous et comment restaurer la santé mentale*.

[3] Personnellement j'ai tout simplement stoppé ma gamme Hippocrate© quand ces progiciels sont devenus des comptas connectées à la carte Vitale. Mon truc c'était l'aide au diagnostic, les statistiques, les interactions médicamenteuses, etc, pas la compta… Cela ne m'excitait plus assez pour continuer à éditer des logiciels de ce type. Et puis j'en avais assez de faire des procès à tous les margoulins qui me piquaient le nom de ma marque pourtant déposé à l'INPI.

Dans ce livre que tout Designer et tout informaticien devrait avoir lu[4] (je n'en connais hélas pas de traduction en français), Alan Cooper aborde le délicat sujet de l'interaction homme / machine. Dès le début du livre il enfonce le clou de son thème central qu'il appelle la "*cognitive friction*", qu'on peut traduire par "friction cognitive" mais ce qui ne vous avancera pas beaucoup :-) !

Il définit cette expression de la sorte :

> "*…the resistance encountered by human intellect when it engages with a complex system of rules that change as the problem permutes*."

Ce qu'on peut traduire rapidement par

> "*la friction cognitive est la résistance rencontrée par l'intellect humain quand il est confronté à un système complexe de règles qui changent en même temps que le problème évolue*".

Le livre étant assez ancien, Alan Cooper prenait beaucoup de références dans les appareils high-tech de l'époque, certains ont évolués ou n'existent plus, mais le raisonnement reste valable sur les appareils d'aujourd'hui hélas (et encore plus sur les logiciels) !

Dans sa définition de la friction cognitive il veut insister sur la stupidité qui préside à la conception de certains appareils high-tech, comme les magnétoscopes, les micro-ondes et, bien entendu, les logiciels, se servant d'analogies avec les premiers pour mieux faire comprendre comment on retrouve les mêmes travers dans ces derniers.

Le principe est simple : si vous prenez un violon, aussi difficile qu'il soit d'apprendre à en jouer, vous n'arriverez jamais à un "méta état" dans lequel les "entrées" que vous donnerez au violon le feront sonner comme un cor de chasse, une cloche ou un saxophone. Alors que si vous prenez un magnétoscope (de l'époque donc) les touches changent de sens selon la fonction : les utilisateurs en viennent à conclure que les petits boutons en plastique doivent coûter une fortune, puisqu'il n'y a que quelques malheureux boutons pour faire des dizaines de choses… Par leur nombre limité, les boutons prennent une signification différente selon "'l'état'" dans lequel se trouve l'appareil. C'est pourquoi chez beaucoup de gens (votre grand-mère, la mienne, chez le vieil oncle Robert…) le magnétoscope affiche toujours glorieusement un "12:00" clignotant : comprendre comment régler cette fichue horloge à chaque fois que le courant disjoncte est finalement bien trop compliqué que le petit avantage d'avoir l'heure à cet endroit procure.

Bien entendu le livre date un peu, et tous les magnétoscopes récents règlent leur horloge automatiquement désormais. Et pourquoi ? Parce que jamais elle n'était réglée correctement par les utilisateurs… Et pourquoi ? Parce ces quelques boutons changeant de sens selon le contexte sont un

[4] Alan Copper a aussi écrit « About Face : The essentials od Interaction Design », un excellent bouquin qu'on peut encore trouver, en anglais aussi.

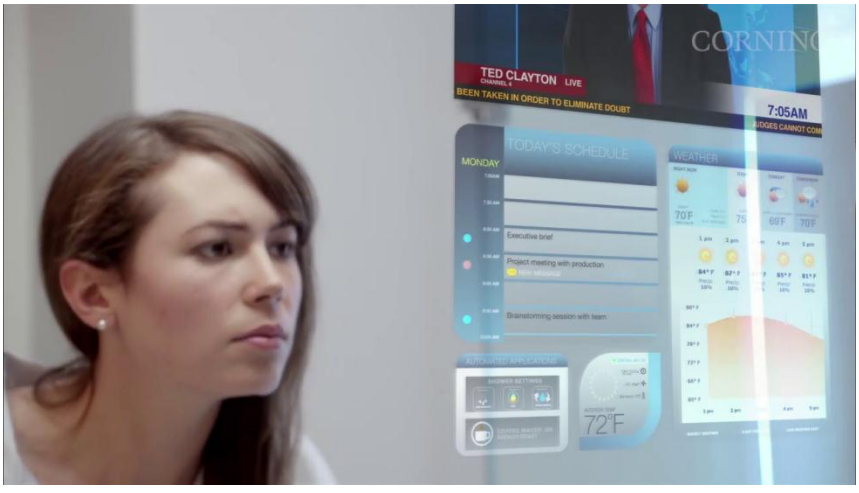
enfer ! Les autres fonctions, notamment la programmation des enregistrements passent aussi aujourd'hui par des commandes visuelles écrites sur l'écran. C'est censé être plus simple. Mais là aussi on rencontre beaucoup de « cognitive friction » !

Prenez certains micro-ondes, si vous n'avez pas le manuel sous les yeux, à quoi correspond le mode "fc-2" par rapport à "fc-3" ? ou "sc-1" et "sc-2". J'ai pourtant un micro-onde dernière génération qui fait four, chaleur tournante et tout le reste, et dont le prix était vraiment élevé, et pourtant en face avant, un mini afficheur à led ne donne que ce genre d'information ! Depuis des années que je l'ai, je suis obligé d'avoir le manuel (qui commence à partir en morceaux) à côté, car sinon, fini la cuisson automatique en mode sc-4 qui utilise la grille du bas mais pas celle du haut et la chaleur tournante et les micro-ondes à 360 W, ce qu'il ne faut surtout pas confondre avec le mode sc-3 qui lui utilise la grille du haut… Alan Cooper est un visionnaire, bien avant que des micro-ondes aussi complexes n'existent il en parlait dans son livre ! Et rien n'a changé, tout a empiré…

La "cognitive friction" est une souffrance, elle créée le rejet. Les logiciels qui sont conçus comme tous ces appareils high-tech sont des atrocités pour l'utilisateur ! Si certains de ces utilisateurs devenaient Empereur, il est certain que l'élu en question commencerait par envoyer au peloton d'exécution tous les informaticiens ayant conçu les logiciels de ce type qu'il aurait dû utiliser durant sa vie ! (personnellement je proposerai quelques noms ☺).

Programmer un magnétoscope est une opération qui fut très longtemps (et le reste parfois encore) une expérience pleine de frustration et de "cognitive friction". Jouer du violon ou du piano, jamais. Et c'est un musicien qui vous le dit. C'est dur. Parfois très dur. Mais jamais on ne connaît de cognitive friction aussi frustrante que celle que créent certains logiciels ou appareils électroniques. **Jamais**.

Peu importent les objets, et peu importe si certains exemples de Cooper ont pris de l'âge en peu de temps, ce qu'il exprime est essentiel, et *comprendre comment créer des logiciels qui n'entraîneront pas de "cognitive friction" est vraiment un besoin urgent, actuel et totalement moderne, au sens d'actuel !*



Cela nous ramène bien entendu aux interfaces utilisateurs. Et pourquoi ? Parce que dès le début je vous ai expliqué que c'est la seule chose "palpable" que

l'utilisateur verra de votre travail ! Son seul contact avec votre code, son seul moyen de communiquer avec le logiciel.

Pensez-y, car les logiciels n'ont rien à voir avec les magnétoscopes : ils sont des centaines de fois plus complexes encore, proposent des dizaines, des centaines d'états internes et externes différents, réagissent différemment selon ces contextes. Les logiciels, plus ils se veulent "souples" plus ils créent de la friction cognitive. Tout le monde connaît au moins un logiciel "standard" dont le paramétrage est tellement sophistiqué qu'il faut vendre une formation ou envoyer un professionnel pour adapter le logiciel aux besoins de l'utilisateur. Un bel exemple de ce qu'il ne faut pas faire…

Les conséquences de la friction cognitive

La friction cognitive entraîne aussi son lot de problèmes. Par exemple TOUS les utilisateurs (même les plus geeks) n'utiliseront généralement qu'un sous-ensemble des possibilités du logiciel. Si on prend par exemple les produits phares de Adobe (PhotoShop et Illustrator), à chaque fois que je m'en sers je ferai rôtir en enfer les informaticiens pervers qui ont créé ces monstres. Essayer de dédresser une photo avec PhotoShop… Et essayer avec PaintShop Pro. Vous comprendrez ce que je veux dire.

> ***Bien entendu cela ne veut pas dire qu'il faut supprimer des fonctionnalités aux logiciels !***
> ***Non, mais en revanche cela oblige à mieux concevoir leurs interfaces !***

Parmi les conséquences de la friction cognitive il y a quelque chose de terrible : l'utilisateur se sent idiot. Et quand une chose créée ce genre de sentiment, en toute logique vous la fuyez. Personne n'aime passer pour un idiot. *On choisira donc de préférence des logiciels plus simples, n'offrant pas certaines possibilités ou une certaine "souplesse" mais dont l'utilisation semblera simple et évidente*. Devinez pourquoi tout un tas de gens achètent des Apple au lieu d'un PC ? … Imaginez un utilisateur de base devant XP qui, pour fermer l'ordinateur réclame de cliquer sur le bouton … "démarrer" !!! Le virage à 180° pris par Microsoft depuis le début de l'ère .NET était essentiel. Tardif, mais essentiel. Et c'est pourquoi aujourd'hui Microsoft met les bouchées doubles pour offrir à la fois des OS et des outils de développement qui ne créent pas de friction cognitive. Mais durant des années ils ont laissé la porte ouverte à Apple qui depuis le Mac avait compris qu'on n'éteint pas un logiciel en cliquant sur « démarrer ».

Je pourrais vous citer encore un autre exemple très simple : Prenez la « Table de caractères » de Windows 7, dernière version donc ayant pourtant subi tout le relooking de Vista et le poids de la prise de conscience par Microsoft de cet impératif de faire des UI intelligibles. Avez-vous utilisez la « Table de caractères » ? Elle n'est vraiment utile que dans un cas : les polices de caractères spéciaux, comme Wingdings. Chercher le symbole que vous voulez, certes vous pouvez faire un copier / coller, mais selon le logiciel que vous utilisez, vous voulez utilisez la fonction Windows qui permet de taper le code (ALT+code) pour l'insérer dans un texte, un logiciel de dessin par exemple qui ne gère par le copier/coller de fontes comme Word. Vous avez juste besoin du code ou de la touche équivalente… Et d'une le code dans la touche n'est pas indiqué, et de deux, oui, aussi ahurissant que cela puisse paraître, le code est indiqué en tout petit dans la barre d'état en… hexadécimal, formaté façon C,

alors même que la fonction (ancestrale) de Windows permettant de taper le code au clavier réclame un code en décimal !

Cet outil pourtant indispensable de l'OS n'a pas évolué d'un poil en 15 ans. Comment, par exemple, blâmer un graphiste, qui utilise de nombreuses fontes, d'avoir acheté un Mac et non un PC sous Windows…

Hélas, Windows, même en version 7, est encore plein de telles bêtises qui, visiblement, ne choquent pas les informaticiens, ceux de chez Microsoft, et ceux que je croise tous les jours dans mes prestations d'audit, de conseil ou de formation… Je comprends dès lors fort bien pourquoi des gens comme moi sont obligés de ramer pour expliquer les fondamentaux du Design dans le monde Microsoft. Ce que je dis aujourd'hui, les informaticiens travaillant sur Mac ou IPad le savent depuis toujours. Autre culture…

Autre conséquence de la friction cognitive : le coût exorbitant de la maintenance et du support. Vous allez me dire certains en vivent… Je connais des sociétés ou des indépendants qui vivent juste sur les formations et le paramétrage de certains logiciels "standard"… Quelle que soit la sympathie que j'ai pour certains d'entre eux, il s'agit d'une hérésie.

Enfin, l'acmé de la friction cognitive est atteinte par les informaticiens eux-mêmes. Lorsque leur logiciel est incompréhensible, lorsqu'il créée de la friction cognitive chez l'utilisateur c'est que c'est ce dernier qui est un idiot, un inculte, un paresseux qui n'a pas lu la documentation, un crétin des alpes. D'ailleurs ce sont les mêmes informaticiens qui tenteront de vous démontrer que leur paramétrage infernal est une nécessité absolue parce que, disent-ils le torse bombé par l'orgueil, leur logiciel est "souple" (ou "évolutif", ou "personnalisable") !

> ***N'ayez pas cette arrogance de pisseur de lignes qui ne mérite que le chômage de longue durée : si l'utilisateur n'y comprend rien c'est que votre interface est bonne à jeter à la poubelle, c'est tout.***

Les "interfaces utilisateur" du monde réel

Le monde réel nous oblige à de constantes interactions avec des objets de toute nature, avec notre environnement, qu'il soit totalement naturel ou bien artificiel, ou entre les deux. Cela inclut les humains qui s'inscrivent dans "le décor".

Certaines choses sont simples, d'autres sont complexes. Certains objets sont durs à utiliser, d'autres se manipulent sans même y penser. Choses et objets au sens le plus large.

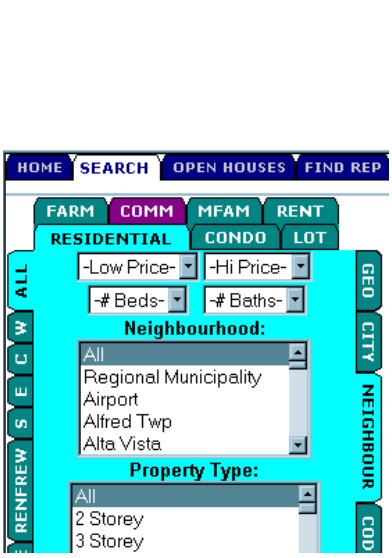
Et cela s'applique aussi aux êtres humains !

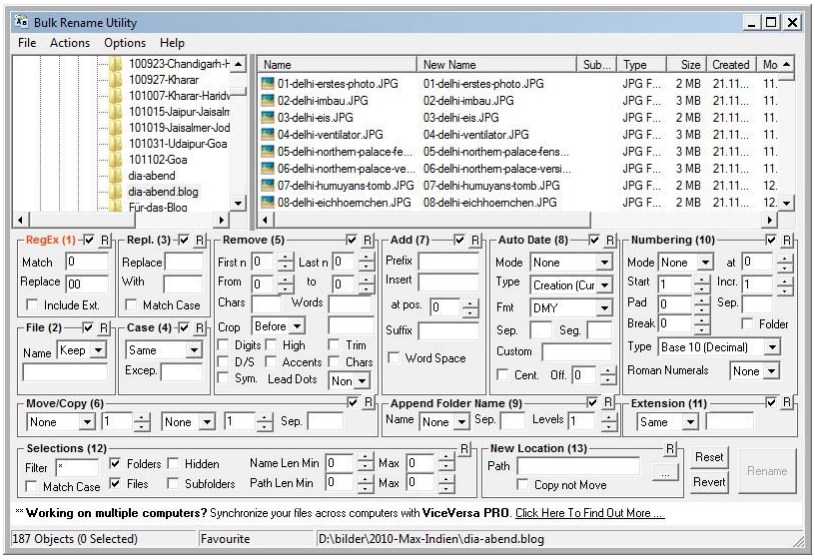
Il en résulte que les concepteurs d'interfaces utilisateur sont souvent de bons communicants. Je parle bien de Designers créant des UI. Pas des infographistes qui, bien que parfois talentueux sont rarement de bon communicants, tout comme les informaticiens d'ailleurs.

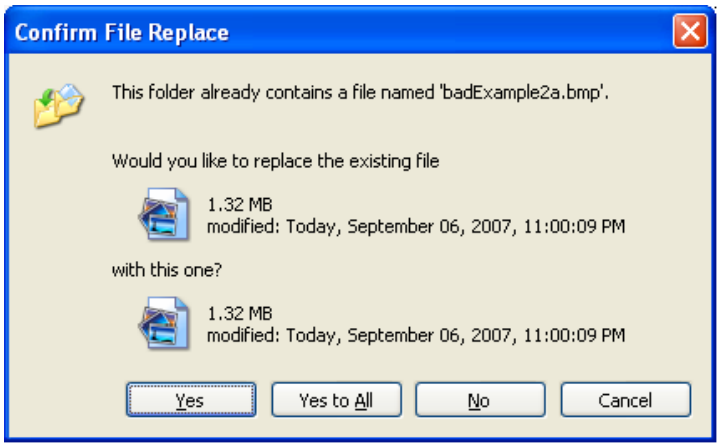
Dans la même logique, si vous améliorez votre communication, vous deviendrez certainement meilleurs pour créer des interfaces utilisateurs !

Voici quelques exemples « d'interfaces utilisateur » puisées du monde réel qui nous entoure :

Quelques exemples de friction cognitive en informatique :







"L'apprentissabilité"

Quel horrible mot ! Mais je n'en ai pas trouvé de plus simple pour faire comprendre le concept…

L'une des questions qu'il faut se poser sur une application c'est de *savoir si l'utilisateur peut rapidement la découvrir et apprendre à s'en servir.*

On peut se dire que cela revient un peu à la fameuse "première impression" dont j'ai déjà parlé. En quelque sorte oui, d'apparence. Dans les faits il s'agit bien de deux choses différentes.

La première impression, comme son nom l'indique, n'est qu'une… impression. Nous nous sommes tous "fait avoir" un jour par un produit (peu importe sa nature) qui "semblait" bien conçu, simple, mais qui, à l'usage, s'est révélé décevant (et je ne parle pas des humains !).

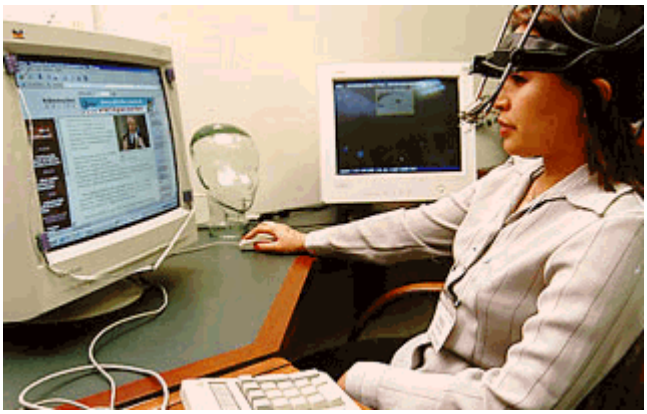
Il a beaucoup de logiciels développés dans cet esprit. Ils sont généralement produits par des sociétés dirigées par des commerciaux… Tout dans la pub, l'apparence, le clinquant, mais pas de budget pour ce "qu'il y dedans". C'est navrant mais cela existe. Je ne vous parle donc pas ici de ce genre de chose qui, à mon sens, frise l'arnaque.

Je vous parle plutôt d'un logiciel faisant bonne impression dès le départ, mais qui à l'utilisation se révèle facile à prendre en main, accompagnant l'utilisateur dans son apprentissage, ne le faisant jamais passer pour un idiot, et même, si possible, lui donnant l'impression d'être plus intelligent qu'il ne l'est en réalité !

Car là est la clé du succès : si votre logiciel donne l'impression à l'utilisateur qu'il s'en sort facilement, alors qu'il avait peut-être une certaine appréhension, alors il vous aimera, vous bénira ! Lui qui avait peur de se sentir un peu bête, se rend compte qu'il comprend tout… Si vous arrivez à créer ce sentiment chez l'utilisateur alors votre logiciel se vendra tout seul par le bouche à oreille !

Mesurer "l'apprentissabilité"

Il existe de nombreuses études portant sur le sujet, souvent très universitaires et peu diffusées. Il existe aussi des cabinets d'experts en ergonomie capables de mesurer avec précision les mouvements des yeux des utilisateurs découvrant votre logiciel ou votre site Web. Le prix de leurs prestations est généralement assez dissuasif, seules de très grosses sociétés pouvant s'offrir de tels services.



Mais il n'est pas interdit de faire dans la simplicité : réussir à trouver deux ou trois utilisateurs types du logiciel, les mettre devant l'écran et leur demander d'exécuter une tâche précise et habituelle pour eux (pour un comptable saisir une note frais par exemple). Prenez une montre et mesurez combien de temps cela nécessite. Testez deux ou trois options différentes de l'interface. Comparez les résultats.

> ***Si vous n'êtes pas en mesure de réunir deux ou trois utilisateurs potentiels de votre logiciel, alors laissez tomber la création d'applications. Franchement. Car si vous ne pouvez pas réunir quelques utilisateurs potentiels comment avez-vous fait pour connaître le profil précis et les habitudes de ces mêmes utilisateurs alors qu'il s'agit d'un préliminaire incontournable ?***

Vous le voyez, **l'échec s'inscrit souvent dès le départ** : une conception coupée des utilisateurs ne pose pas seulement que des problèmes d'interface, mais de cahier des charges, de fonctionnel aussi… ***Un logiciel conçu dans l'éloignement le plus total des utilisateurs ne pourra jamais être un bon logiciel***. C'est ainsi. A la trappe l'idée du « génial inventeur » créant un super soft tout seul sur son portable posé sur la table de la cuisine !

Mais admettons que ce point essentiel est compris et que vous avez eu à cœur d'être proche des utilisateurs dès l'analyse, dès le cahier des charges, et que maintenant que vous en êtes à la conception de l'interface il vous est donc facile de revoir les mêmes gens et de leur demander de faire quelques tests. Supposons que vous puissiez faire ces mesures, ces comparaisons pour aider à prendre de meilleures décisions.

Il n'en reste pas moins vrai qu'au moment de la prise en main l'utilisateur ne sera qu'un débutant, découvrant tout. Il réclamera donc un **support permanent**, ce que le logiciel doit être capable de fournir de lui-même. Vous ne serez pas à côté de tous les utilisateurs pour leur donner des conseils…

D'où l'importance d'intégrer des dialogues supplémentaires, des explications, des info-bulles, des choix du type "débutant / expert" modifiant l'apparence et offrant plus ou moins d'options, sans oublier les "experts" ou "wizards" permettant de réaliser une tâche donnée en étant guidé. N'oubliez pas qu'*un bon "expert" ne doit pas être une boîte noire*, ce qu'on voit trop souvent ! Un bon expert est conçu pour guider l'utilisateur vers son but, mais *en même temps il doit être capable de lui faire comprendre comment atteindre ce but en se passant de l'expert* !

Un logiciel dont "l'apprentissabilité" est excellente répondra au minimum à 80% des besoins de l'utilisateur.

Arrivé à ce point de véritable "souplesse" (et non pas la fausse souplesse-excuse des usines à gaz), vous aurez développé un vrai logiciel moderne. L'intégration de la conception d'une interface utilisateur pensée, avec l'aide d'un vrai Designer, ne vous apparaitra plus alors comme une "mode", un "gadget". C'est que vous aurez compris ce que doit être un logiciel bien conçu de nos jours…

"L'utilisabilité"

Avec "*apprentissabilité*" je ne suis plus à un néologisme abominable près hein…

L'utilisabilité n'existe pas en français, mais cela existe dans la réalité. Concernant ce point important il convient de réfléchir à la question suivante :

A quel point l'interface utilisateur est-elle intuitive et productive pour un utilisateur "moyen" ?

Cela peut aussi se mesurer. Cette fois-ci en prenant pour cobaye un utilisateur avancé et en lui demandant, toujours la même chose, d'effectuer une tâche habituelle. Bien entendu on prendra plutôt deux utilisateurs, voire quatre ou cinq, pour que les mesures aient un sens.

Une interface utilisateur ayant une haute “utilisabilité” permet aux utilisateurs d’effectuer toutes les tâches courantes de façon directe et sans détour. Cela implique dans certains cas que le workflow puisse être personnalisé.

Il faut faire attention à ne pas confondre “utilisabilité” et “apprentissabilité”. La confusion est fréquente car l’un va souvent avec l’autre, mais il s’agit bien de deux domaines distincts. Deux objectifs différents qui ont leurs impératifs propres et leurs mesures distinctes.

Les tests d’utilisabilité réclament un panel de testeurs le plus large possible pour pouvoir tirer des conclusions. Les évolutions, les variantes, tout cela doit être mesuré spécifiquement et séparément. C’est vraiment une partie difficile dans la conception d’un logiciel car cela peut coûter très cher à organiser. On comprend facilement pourquoi les plus grands éditeurs du monde préfèrent généralement sortir des versions “bêta” … A défaut d’un test encadré scientifiquement, les “bêta” permettent de s’offrir un large panel d’utilisateurs pour un coût ridicule ! Les plus malins diffusent durant quelques jours une bêta offrant certains choix d’interface, et pendant les jours suivant une version offrant un choix différent. Tout le monde croyant télécharger la même version bêta bien entendu. En fonction des retours du premier et du second groupe vous saurez quel choix est le meilleur…

La clarté

Enfin un mot qui existe ! Et pourtant celui-là, ironie du sort, son orthographe ne m’a jamais paru claire ! J’ai toujours l’impression qu’il manque un “e” entre le “r” et le “t”. Comme quoi le mot n’est pas la chose qu’il désigne (Cf. « Ceci n’est pas une pipe » de Magritte dans le même esprit). L’orthographe est d’ailleurs, dans notre langue, un excellent exemple de friction cognitive ! Beaucoup de règles qui changent selon le contexte avec tout autant, si ce n’est plus, d’exceptions…

Beaucoup d’interfaces surchargent l’utilisateur de données qui n’ont pas toute la même pertinence. Il est en réalité essentiel de viser une conception « pacifiée », « calme ». Le look « Metro » que Microsoft a produit pour Windows Phone 7 et qui sera ré exploité sous Windows 8 est un exemple d’une telle recherche de « calme ». Circulation du blanc ou du noir pour aérer la présentation, des objets aux formes simples,

reconnaissables, que l'œil repère, classe, trie facilement. Des fonds évocateurs de nature, de paix intérieure. Metro n'est pas l'exemple parfait, mais c'est une des façons d'appliquer de bonnes règles de conception visuelle.

Il faut aussi guider l'utilisateur en créant des groupes visuels qui ont un sens. Les groupes peuvent être logiques ou fonctionnels. Les horizontales doivent être exploitées pour tout ce qui est en rapport avec un espace, une durée, les verticales pour ce qui est apparenté à des énumérations, que ces verticales et horizontales soient matérialisées ou uniquement suggérées par la mise en page (je n'aborderai pas ici les principes de design en pratique, mais c'est ce qu'on appelle les « closures », créer des formes sans les dessiner).

> ***La règle d'or en la matière est qu'il est bien plus important de se concentrer sur la clarté du visuel que d'ajouter des infos-bulles de partout pour expliquer ce qu'on ne comprend pas tout de suite !***

La liberté d'action

Avoir le choix, avoir une grande liberté d'action est une chose essentielle, l'utilisateur ne doit jamais se sentir « piégé » dans une procédure dont il ne peut plus s'échapper.

Mais il y a plus important encore : la liberté de pouvoir changer le comportement du logiciel. C'est la fameuse « souplesse » qu'on retrouve dans des logiciels aux paramètres pléthoriques, mais mise en scène correctement.

Selon le niveau de compétence de l'utilisateur, certaines options sont ou non importantes, et l'accès même à ces options suit cette logique.

Par exemple, un débutant (sur le logiciel considéré) n'aura pas besoin d'avoir accès à de nombreuses options. Il doit pouvoir travailler immédiatement et réaliser les manipulations qu'il souhaite, sans friction cognitive et sans avoir à répondre à des listes d'options dont il ne comprend pas (encore) le sens.

Je vois souvent des logiciels qui, au moment de l'installation (ou de la première utilisation) affichent de nombreux choix qui n'ont pas encore de sens pour l'utilisateur. Comment décider de la mise en page par exemple avant même d'avoir testé celles qui sont proposées ? Visual Studio pose ce genre de question, c'est une erreur. Quelle différence y-a-t-il entre la mise en page « VB.NET » et « C# » ? Vous le savez, vous ? Ne travaillant pas en VB.NET je n'ai jamais choisi cette option et après des années de pratique de VS, je ne sais toujours pas ce qui se passerait si je la choisissais ! Ahurissant ! Du coup je choisi toujours « environnement C# » mais il est fort possible qu'une autre option soit mieux adaptée à mes goûts et ma façon de travailler. Hérésie et Frustration sont les conséquences d'un mauvais Design…

La gestion des options est essentielle : un bon logiciel est souvent paramétrable pour s'adapter aux préférences de l'utilisateur. Mais il doit le faire de façon progressive, graduée, en accord avec les connaissances de l'utilisateur, son habilité.

Cela passe aussi par une phase délicate pour le développeur : choisir les bonnes valeurs « par défaut » pour le débutant.

Il faut se convaincre que si le développeur ne sait pas faire ces choix, alors l'utilisateur débutant le sera généralement encore moins !

Toutes les options doivent être regroupées par thématique, être facilement accessibles et surtout être compréhensibles.

Prenez les options d'Internet Explorer, surtout IE9… C'est un enfer. Il y a trop de lignes pour la petite boîte qui les affiche, aucun moyen de chercher une option, la plupart ont des noms qui n'aident pas à « réellement » savoir ce que cela va faire, et encore je suis informaticien ! Encore un exemple parfait de ce qu'il ne faut pas faire ! Les options de Visual Studio sont déjà mieux structurées, mais ne sont pas toujours plus parlantes, sans parler de l'affreux système de paramétrage du clavier et des commandes.

Comme quoi, il n'y a pas besoin d'aller chercher bien loin les mauvais élèves[5] !

Revenons aux options, en général. Il faut se mettre dans la peau de l'utilisateur. Certains choix deviendront évidents à un utilisateur expérimenté. Il saura où trouver les options « avancées » et les manipuler sans problème. Pour cela le logiciel doit avoir été correctement pensé et offrir différents niveaux d'options.

Utiliser les bonnes métaphores

L'art de la métaphore, comme de la parabole est une sorte de sixième sens. Vous l'avez ou non. Si vous ne l'avez pas, faite appel à un Designer compétent qui saura trouver les métaphores qui font sens.

Les métaphores dans les interfaces sont essentielles car elles font venir une partie du monde réel connu dans le logiciel. Elles introduisent un sens de familiarité entre l'application et l'utilisateur.

Une métaphore réussie se suffit à elle-même, elle supprime le besoin d'un libellé, celui d'une bulle d'aide. Elle peut se passer totalement de texte.

La plus célèbre et parmi les plus réussies, est certainement la métaphore de la corbeille. On comprend immédiatement ce que c'est et à quoi cela sert. Dans le pire des cas on apprend très vite à quelle fonction du logiciel elle fait référence, et on sait s'en servir en quelques secondes.

D'autres métaphores sont tout aussi connues et utilisées régulièrement : la notion de « bureau », de « répertoire », de « document » par exemple.

Il ne faut pas abuser des métaphores. D'abord parce qu'elles sont loin d'être universelles. Il est plus facile de traduire un bout de texte pour internationaliser un logiciel que de refaire toute l'interface en prenant soin d'utiliser des métaphores en phase avec la culture de chaque pays…

[5] A noter que si je « tape » volontiers sur la tête de Microsoft, c'est tout simplement parce que j'en connais bien les produits car j'en suis globalement satisfait. Nul doute, et que le lecteur en soit convaincu, que si je travaillais sur Mac j'aurai certainement autant d'incohérences à vous livrer. J'ai eu un Mac G3, et de cette simple expérience j'aurai déjà toute une liste d'âneries et de trucs générant de la friction cognitive à vous livrer !

Le danger étant qu'une métaphore « convenable » ou « adaptée » dans un pays, une culture, devienne au mieux « curieuse » voire choquante dans une autre culture !

Prenons un petit exemple. En Italie se tapoter le lobe de l'oreille avec l'index signifie (en général en connivence avec son interlocuteur) que la personne dont on parle, ou qui vient juste de passer, est un homosexuel. Dans les régions très au sud, comme la Sicile, au machisme légendaire, un tel geste est donc porteur d'un sens très péjoratif.

Imaginons un instant que vous ayez créé un logiciel qui diffuse du son et que vous ayez tourné quelques petites vidéos pour expliquer les fonctions. On pourrait supposer une séquence où l'acteur se tapote l'oreille comme je l'ai expliqué pour signifier « j'entends mal, comment montez le son ? ».

Je vous laisse comprendre ce qu'une telle métaphore aurait comme effet lorsque vous diffuserez votre logiciel en Italie !

Refaire toutes les vidéos d'aide de votre logiciel risque d'être coûteux, bien plus que de traduire un texte. Plus grave : encore faudra-t-il être capable de déceler ces « anti-métaphores » pour chaque pays où vous diffuserez votre logiciel…

Vidéo, photos, dessins, animations, toutes ces manifestations visuelles peuvent avoir un sens déplacé dans les cultures que vous ne maîtrisez pas. Ne l'oubliez jamais !

Mais il y a aussi les métaphores qui « poussent le bouchon trop loin » notamment en se voulant tellement proches de la réalité qu'elles en deviennent néfastes pour l'application.

Prenons un exemple simple d'un domaine que vous ne connaissez peut-être pas, mais vous allez comprendre. Dans le monde des synthétiseurs modulaires, ceux des années 70/80, le Moog modulaire par exemple, toute la modularité tenait dans le fait que ces appareils étaient dotés de dizaines, voire centaines de prises Jack femelles et qu'à l'aide d'un cordon Jack Mâle/Mâle on pouvait établir des connexions entre les différents modules. Pour avoir connu cette époque je peux vous assurer que c'était le summum du nec plus ultra du jeune geek ! Je garde d'ailleurs

religieusement deux cordons de mon dernier système modulaire… C'est pour dire. Bref, un modulaire en cours de fonctionnement ressemblait aux anciens (très anciens) standards téléphoniques des années 50. Kitch et geek (mais il fallait parfois des heures pour fabriquer un son différent, difficile à utiliser sur scène !).

Depuis que la puissance des ordinateurs a rendu la chose

possible, de très nombreux synthétiseurs « cultes », comme le Moog Modulaire, ont été simulés en numérique pour être utilisés dans des séquenceurs ou arrangeurs (Cubase par exemple). Certains éditeurs ont été tellement loin dans le réalisme, qu'on peut brancher des cordons Jack comme sur le « vrai » synthétiseur… Réalisme amenant au même problème : quand de très nombreux cordons sont branchés on ne voit plus rien de la face avant du synthétiseur, on ne comprend plus quel cordon va où… Les cordons sont si bien simulés qu'ils bougent et se tortillent comme des vrais. Une horreur ! La capture écran enchâssée dans ce texte vous montre un autre logiciel musical utilisant la même métaphore ultra réaliste des cordons…

Cela « jette » en démonstration. Mais à l'usage c'est plus que pénible. Mauvais choix de métaphore, mauvaise balance entre la reproduction du monde réel et la fameuse utilisabilité du logiciel que j'évoquais plus haut.

Le musicien veut une reproduction parfaite des possibilités et du son de la machine originale, mais pas de ses enquiquinements ! J'attends le jour où certains pousseront la plaisanterie jusqu'à obliger l'utilisateur à attendre presque une heure devant son écran avant de se servir de son synthétiseur… En effet, ces machines étaient analogiques et la « mise en température » était longue. Si vous faisiez de l'enregistrement multipiste, il fallait être sûr que tous les composants soient chauds avant de faire le « La », sinon la fréquence bougeait de trop pendant la session de travail et votre morceau de musique final sonnait désaccordé. Le choix des cordons qui gênent la vue et rendre peu lisibles les branchements effectués est aussi stupide que le serait de simuler ce temps de mise en température des vraies machines de l'époque… *La métaphore ne doit pas reproduire les faiblesses du monde réel*.

En matière de métaphore il faut donc avoir la main légère et le coup de crayon précis. Parvenir à l'universalité est impossible ou presque.

Le réglage du thème de Windows ou celui de l'image de fond sont en revanche des métaphores réalistes tout à fait réussies. On comprend tout de suite qu'on regarde une simulation de son propre écran tel qu'il sera si on applique les changements.

Représenter un écran par un écran est une métaphore évidente et facile à mettre en œuvre, sans risque de mauvaise interprétation. Il existe bien plus de cas où trouver la bonne équivalence est un casse-tête. C'est pour cela que l'aide d'un Designer créatif est plus qu'indispensable !

Les utilisateurs ne lisent rien !

Certaines règles dans la fabrication des interfaces sont si évidentes qu'on les oublie… Par exemple tout le monde sait que les utilisateurs ne lisent jamais les messages. Alors pourquoi peut-on encore voir des logiciels afficher des tartines à l'écran ? Ce refus de la réalité est à la limite de la maladie mentale et de l'autisme, il faut faire attention …

La plupart des boîtes de dialogue se ressemble tellement et est tellement utilisée qu'**elles endorment l'utilisateur**. Jusqu'à ce qu'il comprenne qu'il vient par habitude de cliquer sur « oui » mais que la question n'était pas « voulez-vous sauvegarder le fichier » mais « confirmez-vous l'effacement du disque » !

Bannissez les boîtes de dialogue ! Evitez-les le plus possible. Même « relookées », même « jolies », même en accord avec le thème visuel de l'application !

Si l'interface est bien conçue, l'utilisateur ne risque pas de cliquer sur « supprimer » au lieu de « sauvegarder ». Lui demander de confirmer « voulez-vous supprimer… » est stupide. D'autant qu'un bon logiciel doit permettre l'erreur, donc le « undo ». Sinon on en arrive aux dialogues de confirmation incessants de Vista qui ont usé les nerfs des plus patients. Ou au message systématique qu'on se « prend » (tel un poing en pleine figure) à chaque fois qu'on passe une vidéo en plein écran sur Internet, juste pour vous dire … je ne sais plus quoi d'ailleurs ! Preuve de l'inefficacité totale du procédé (comme tout bon utilisateur j'ai juste mémorisé ce qui m'intéresse : il faut taper sur Escape pour sortir de ce mode).

Cela est stupide car l'utilisateur est endormi par tous les dialogues et qu'il y répond machinalement, **donc en annihilant l'effet recherché**, c'est stupide parce que **cela gêne l'utilisateur**, c'est stupide parce que cela « casse le rythme », entrave la fluidité des actions de l'application.

Si votre logiciel est obligé d'afficher un dialogue, concevez-le comme une publicité : le message doit être court, sans équivoque, percutant. Les tournures alambiquées, les questions interronégatives et autres placards d'explications jamais lus sont plus qu'à éviter : considérez cela comme une interdiction totale, absolue, de la même nature qu'il est interdit d'éteindre son ordinateur en arrachant la prise électrique !

La formulation d'un dialogue doit être positive, claire, et donc basée sur le contexte. Si vous affichez le même dialogue dans deux situations différentes dites-vous que vous êtes dans l'erreur !

Bien entendu, comme pour une publicité ou un discours commercial, toutes les tournures ou mots ayant un sens négatif sont à éliminer. Les messages du type « Fonction Interdite ! » sont des abominations. Tout comme terminer les messages d'erreur par des points d'exclamation. Quelle sale manie ! « Imprimante indisponible ! », « Réseau non détecté ! ». Oh la-la … On se croirait face à une maîtresse revêche ancienne école, prête à vous taper sur les doigts avec sa règle en bois !

Les messages, même d'erreur, doivent être ponctués d'un simple point. Nul besoin non plus de sons d'alarme imitant le Nostromo[6] avant son autodestruction ! Ni même d'icônes angoissantes genre tête de mort ou symbole nucléaire !

Un bon logiciel est un logiciel qui a été conçu comme je le disais aussi plus haut pour inspirer le « calme », la « sérénité ». Vous imaginez Maître Yoda se mettre à courir dans tous les sens en poussant des petits cris aigus parce que Dark Vador est dans le secteur ? Franchement ? Non. Un bon logiciel non plus. Il doit donner l'impression à l'utilisateur que tout est sous contrôle. Un « plantage » ? Il s'en gausse ! Tout juste le signale-t-il calmement, sans affoler tout le monde, en ayant pris soin de créer un Log détaillé et de tenter de revenir à la situation précédente sans faire perdre tout le travail de l'utilisateur. Et s'il doit s'arrêter, qu'il le fasse dans la sérénité, en se relançant automatiquement et en tentant de réafficher les informations telles qu'elles étaient avant le plantage.

Ça c'est de la programmation de haute voltige, là il peut y avoir fierté du développeur.

[6] Le cargo spatial dans lequel se déroule le film Alien (le 1).

Conclusion

L'informaticien, celui que je rencontre le plus souvent, est trop souvent hélas, totalement ignorant de ce bouleversement qui tel un tsunami pousse toute la production de logiciels vers le Design d'UI intelligentes et agréables.

Pire, certains se rient de tout cela. Se moquant peut-être de ce qui leur fait peur, de voir leur métier leur échapper, se diriger dans une direction où ils savent, au moins inconsciemment, qu'ils n'ont aucune compétence…

Pourtant le Design n'est pas tant être capable de dessiner la Jaconde que de savoir en imaginer les traits et la posture…

Et tout informaticien peut acquérir les compétences qui lui permettront de concevoir des interfaces ergonomiques et agréables, quitte à s'associer à un vrai Designer ou même un simple infographiste pour l'aider dans la partie purement visuelle, dans le dessin qu'il ne saura jamais réaliser lui-même. Créer un concept ne réclame pas de savoir le réaliser. Croyez-vous que Philippe Starck sache façonner lui-même les objets qu'il conçoit ? Qu'il maîtrise la ferronnerie, l'extrusion des plastiques, l'art du menuisier, du tailleur de pierre, du maître verrier ?

Bien heureusement, j'en croise d'autres qui ont pris conscience de ce mouvement et qui s'y intéressent, même confusément.

Pour certains c'est une simple « curiosité », ils regardent tout cela avec un intérêt détaché comme si « eux » n'étaient pas concernés par ce changement… Mais c'est un bon début !

L'informatique est un outil, un simple outil. Comme une pelle ou un marteau.

La seule différence est dans le fait que les outils classiques permettent d'étendre la force ou l'habilité physique, alors que l'informatique est un outil destiné à étendre la force et l'habilité intellectuelle.

Un marteau est bien plus dur qu'une main, et plus lourd, mieux adapté que celle-ci pour enfoncer un clou. Mais ce n'est qu'un prolongement de la main qui le tient. Mieux, le marteau n'est qu'une main, façonnée par l'esprit de l'homme pour être adaptée à une tâche. Une greffe temporaire.

Un ordinateur est bien plus précis qu'un cerveau humain, il ne se lasse pas des tâches répétitives, là où le cerveau perd sa vigilance par habitude et lassitude. Il n'est lui aussi qu'un prolongement, celui de l'esprit qui l'utilise, une extension de sa mémoire, de ses capacités de calcul.

L'informatique n'est qu'un outil. Tous les outils, depuis le jour de leur création, tous, ont été voués à évoluer, à se perfectionner. C'est dans la nature de l'humain de transformer un caillou en outil pour casser ou tailler d'autres outils, puis de transformer ce caillou en marteau, et enfin d'ajouter à ce marteau une partie gommée sur le manche pour mieux le saisir, d'équilibrer son poids pour qu'il ne fatigue pas le poignet : de l'outil naturel, le caillou, l'humain arrive à un objet composite complexe, utilisant des matériau différents réclamant chacun une maîtrise technologique différente, à une évidente prise en compte du confort de son utilisateur. Pas pour la « beauté » ou l'esthétique, mais d'abord pour lui-même, son propre confort égoïste. Puis celui des autres, parce qu'après tout, l'humain est avant tout un animal social.

Pourquoi cette évidence échappe-t-elle autant aux informaticiens lorsqu'il s'agit des logiciels ?

Soyons lucides. Il est évident que l'informatique ne pouvait, comme tout outil, en rester à ses formes et apparences de ses débuts.

Aujourd'hui, tel le papillon émergeant de sa chrysalide, l'informatique se dépouille de ses écrans verts, de ces grilles de valeurs numériques rébarbatives, de ses lourdeurs, de son manque d'adaptation aux utilisateurs.

Soyez l'entomologiste de ce nouveau monde en prenant soin de ce papillon qui ne réclame que votre aide pour s'envoler…

Bon Développement !

Olivier Dahan
MVP Silverlight

Pour tout conseil, formation, audit ou développement, contact : odahan@e-naxos.com