# Le mythe de la métaphore.

*"The myth of metaphor" un article d'Alan Cooper (Juin 1995) publié sur son site:*
*http://www.cooper.com/articles/art_myth_of_metaphor.htm*
*Traduction: Marc Wathieu*

Les designers de logiciels parlent souvent de «trouver la bonne métaphore» sur laquelle baser la conception de leur interface. Ils imaginent qu'une transposition de leur interface-utilisateur en images d'objets réels et familiers permet une compréhension automatique par leurs utilisateurs. Ainsi, ils traduisent leur interface-utilisateur en un bureau rempli d'armoires à dossier, de téléphones et de carnets d'adresses, en une pile de papier ou encore en une rue bordées de bâtiments, dans l'espoir de créer un programme idéalement compréhensible. Si vous recherchez cette métaphore magique, vous serez en bonne compagnie: certains des plus brillants concepteurs d'interfaces ont fait du choix de la bonne métaphore une priorité absolue.

Mais en recherchant cette métaphore magique vous commettrez une des plus grandes erreurs en conception d'interface. La quête de cette métaphore sacrée est comme chercher le moteur à vapeur idéal pour équiper votre avion, ou chercher le bon dinosaure à chevaucher pour aller travailler.

Je pense que baser une conception d'interface-utilisateur sur une métaphore est non seulement inutile mais peut souvent s'avérer nocif. L'idée qu'une bonne conception d'interface doit être basée sur des métaphores est l'un des plus insidieux parmi les nombreux mythes qui imprègnent la communauté des concepteurs de logiciels.

Les métaphores offrent un bien mince avantage pour l'apprentissage des utilisateurs-novices, par rapport à leurs conséquences énormes. Le plus gros problème est qu'en représentant de vieilles technologies, les métaphores réduisent fermement tout développement conceptuel, limitant à tout jamais le potentiel de notre logiciel. Elles posent également une foule d'autres problèmes: leur nombre n'étant pas infini, leur pertinence est compromise, et la capacité des utilisateurs à les identifier est incertaine. Confondre leur rôle est le problème posé par la plupart de ce que nous considérons à tort comme des interfaces métaphoriques.

## Les trois paradigmes[1] de l'interface.

Je pense qu'il y a trois paradigmes dominants dans les interfaces de logiciel: le *paradigme technologique*, le *paradigme de la métaphore*, et le *paradigme idiomatique*[2]. Le *paradigme technologique* est basé sur la compréhension du fonctionnement des choses: objectif difficile. *Le paradigme de la métaphore* est basé sur le fait de deviner intuitivement comment fonctionnent les choses: méthode problématique. *Le paradigme idiomatique*[2] est basé sur le fait d'apprendre comment accomplir les choses: processus normal et humain. Globalement, nous sommes passés de la technologie à l'usage de la métaphore, et nous nous rendons à présent compte de l'importance de la conception idiomatique. Bien que chacune de ces trois phases soit évidente dans le domaine du logiciel contemporain, le *paradigme de la métaphore* est le seul à avoir été popularisé. Nous lui payons donc un lourd tribu et, trop souvent, nous entravons la création d'interfaces vraiment bonnes par un usage erroné des métaphores.

1. Paradigme: modèle théorique de pensée qui oriente la recherche et la réflexion.
2. Idiomatique: propre à tel ou tel idiome (moyen d'expression propre à une communauté).

## Le paradigme technologique.

Le *paradigme technologique* de l'interface-utilisateur est simple et incroyablement répandu dans l'industrie informatique. Il signifie simplement que l'interface exprime les termes de sa propre construction, la manière dont il a été construit. Afin de l'employer avec succès, l'utilisateur doit comprendre comment le logiciel fonctionne.

Dans les années 60, il y eu un mouvement architectural appelé les *métabolistes*. Son influence se fait encore ressentir. Dans l'architecture *métaboliste*, les axes d'ascenseur, les conduits de climatisation, les gaines de câble, les montants en acier ou tout autre matériau de construction sont laissés à découvert, visibles à la fois de l'intérieur et de l'extérieur. Les muscles, les os et les tendons d'un bâtiment sont exposés - et même soulignés - sans la moindre modestie. L'idée étant que le bâtiment est une machine à vivre et que sa forme devrait suivre ses détails d'exécution. Une écrasante majorité de logiciels sont aujourd'hui métabolistes du fait qu'ils nous montrent sans honte comment ils sont construits: un bouton par fonction, une fonction par module de code, commandes et processus, qui font écho avec précision aux structures et aux algorithmes de données internes.

Nous pouvons voir comment un programme fait tic-tac simplement en apprenant à le parcourir. Le problème est que l'inverse est également vrai: nous devons apprendre comment il fait tic-tac afin de le parcourir. Les ingénieurs veulent connaître le fonctionnement des choses, et le paradigme technologique les satisfait très bien, ce qui explique naturellement pourquoi tant de logiciels l'utilise. Les ingénieurs préfèrent voir toutes les pièces, leviers et autres valves puisque cela leur permet de comprendre ce qui se passe à l'intérieur de la machine. Que ces éléments compliquent inutilement l'interface ne semble pas les gêner. Bien que les ingénieurs veuillent comprendre ce type de fonctionnements internes, ce n'est pas le cas de la plupart des utilisateurs qui manquent de temps pour cela. Ils préfèrent que cela fonctionne efficacement plutôt que de savoir comment cela fonctionne, attitude souvent difficile à comprendre pour des ingénieurs.

## Le paradigme de la métaphore.

Dans les années 70, l'interface-utilisateur graphique moderne a été inventée au *Palo Alto Research Center* (PARC), centre de recherche de la société Xerox à Palo Alto, Californie. Elle a balayé l'industrie; mais de quoi s'agit-il exactement ? Le GUI (Graphic User Interface), comme défini par le PARC, est composé de beaucoup de choses: fenêtres, boutons, souris, icônes, métaphores, menus étirables. Certaines de ces composants sont bons, d'autres le sont moins. Ils ont cependant tous accédé à un statut indéboulonnable au sein de l'industrie, étant indistinctement associés à la qualité de l'ensemble. En particulier, l'idée que les métaphores sont une base solide et indispensable pour la conception d'interfaces est une proposition plutôt fallacieuse. Ce serait un peu comme adorer les disquettes souples sous prétexte qu'à l'époque, elles ont contenu de bons logiciels.

La première réalisation commercialement réussie d'un GUI du PARC était le Apple Macintosh, avec les métaphores du bureau, de la poubelle, de feuilles de papier superposables, de fardes et de dossiers. Le succès des Macs n'était pas dû à ces métaphores, mais bien au fait qu'il fût le premier à définir un vocabulaire restreint permettant de communiquer avec l'utilisateur, sur base d'un nombre réduit d'actions à la souris. Les métaphores étaient juste des peintures gentilles sur les murs d'une maison bien conçue.

Les métaphores ne sont pas extensibles. Une métaphore qui fonctionne bien pour un processus simple dans un programme simple échouera souvent si le processus grandit en taille et en complexité. Les icônes de dossiers étaient une bonne idée quand les ordinateurs avaient des disquettes souples ou des disques durs de 10 Mo.
À l'heure où la capacité des disques durs se compte en Giga-octet et où les dossiers se comptent par milliers, l'usage d'icônes peut parfois ressembler à un joli désastre. Nous comprenons les métaphores par intuition. Dans les interfaces-utilisateur, nous saisissons la signification de la commande métaphorique parce que nous la relions mentalement à un autre processus ou à autre chose que nous avons appris au préalable. La grande force de cette méthode est son efficacité, tirant profit de la puissance impressionnante de l'esprit humain à faire des inférences[1], choses dont les CPUs (*Central Process Units ou Unités Centrales de Calcul*) sont incapables. La faiblesse de cette méthode est donc qu'elle dépend d'un esprit humain qui peut se montrer grinçant, revêche, idiosyncratique[2], ou qui ne possède parfois pas la connaissance ou la puissance déductive nécessaire pour établir la bonne connexion. Les métaphores ne sont pas aussi sûres que notre compréhension. Parfois la magie opère, parfois pas.

L'intuition du *paradigme de la métaphore* a lieu sans nécessité de comprendre les mécanismes d'un logiciel. Il s'agit ainsi d'un pas en avant par rapport au paradigme technologique, mais sa puissance et son utilité ont été gonflées en proportions peu réalistes. Le Larousse définit l'intuition comme une «forme de connaissance immédiate qui ne recourt pas au raisonnement». Wow ! Aucune pensée impliquée. Il serait idiot d'imaginer que nous puissions baser une bonne conception d'interface-utilisateur sur un genre de magie mentale qui excluerait la pensée. Nous comprenons intuitivement les choses par comparaison mentale immédiate avec ce que nous avons déjà appris. Vous comprenez intuitivement ce que signifie l'icône de la poubelle parce que vous avez un jour fait l'effort d'apprendre la fonction d'une vraie poubelle, préparant ainsi votre esprit à en comprendre dorénavant le fonctionnement. Mais en son temps, vous n'auriez pas pu comprendre intuitivement l'usage d'une poubelle. Il était juste extrêmement facile de l'apprendre. Ce qui nous amène au paradigme idiomatique, qui est basé sur le fait que l'esprit humain est une incroyable et puissante machine à apprendre, et que l'apprentissage n'est pas dur pour nous.

**Le paradigme idiomatique.**

Cette troisième méthode de conception d'interface-utilisateur résout les problèmes des deux précédents. Je l'appelle *idiomatique* parce qu'elle est basée sur la manière dont nous apprenons et employons des idiomes, ou figures de langage, comme le «c'est trop de la balle !» ou «nickel !» Elles sont facilement comprises mais pas de la même manière dont les métaphores le sont. Il n'y a là ni balle, ni métal. Nous comprenons l'idiome parce que nous l'avons appris et parce qu'il est distinctif. Simple, non ? C'est là où l'esprit humain est vraiment exceptionnel, maîtrisant l'apprentissage et se rappelant des idiomes très facilement sans devoir les comparer à des situations connues ou comprendre comment ils fonctionnent. Il en est ainsi car la plupart des idiomes n'ont aucune signification métaphorique possible. La plupart des commandes sur une interface de GUI sont des idiomes. Les pointeurs, les réducteurs de fenêtres, les touches d'un player, les boîtes flottantes et les poignées (*scrollbars*) sont des choses que nous apprenons «*idiomatiquement*» plutôt que «*métaphoriquement*» ou intuitivement.

1. Inférence: opération logique par laquelle on admet une proposition en vertu de sa liaison avec d'autres propositions tenues pour vraies.

2. Idiosyncratique: réagissant d'une façon personnelle.

Nous tendons à penser que tout apprentissage est dur en raison du conditionnement du *paradigme technologique*. Il était très difficile d'apprendre ces vieilles interfaces-utilisateur parce que vous avez dû également en comprendre le fonctionnement. La plupart de ce que nous connaissons, nous l'apprenons sans compréhension: des visages, des interactions sociales, des attitudes, l'arrangement des pièces et des meubles dans nos maisons ou nos bureaux. Nous «ne comprenons pas» pourquoi tel visage est comme il est, mais nous «connaissons» ce visage. Nous l'identifions parce que nous l'avons regardé et l'avons appris par coeur, et ce n'était pas difficile.

Notre souris familière n'est pas métaphorique de quoi que ce soit mais est plutôt apprise idiomatiquement. Cette scène dans *Star Trek* IV où Scotty revient sur terre au XXIe siècle et essaie de parler dans une souris est un des quelques moments de ce film qui ne sont pas de la science-fiction. Il n'y a rien qui puisse indiquer le but ou l'utilisation d'une souris, ni de comparable à toute autre chose dans notre expérience: son apprentissage n'est pas intuitif. Cependant, apprendre à pointer quelque chose avec la souris devient incroyablement facile. Quelqu'un a probablement passé trois secondes à vous le montrer la première fois, et vous l'avez maîtrisé en un instant. Nous ne savons pas ou ne nous inquiétons pas de comment la souris fonctionne mais nous pouvons l'actionner parfaitement. C'est l'apprentissage *idiomatique*.

L'observation-clé au sujet des idiomes est que, bien qu'ils doivent être appris, ils ne doivent l'être qu'une seule fois. Il est tout à fait facile d'apprendre des idiomes comme «grunge» ou «glander» ou «ça déchire». L'esprit humain est capable d'assimiler un de ces idiomes en une seule audition. De la même manière, il est facile d'apprendre des idiomes comme des cases à cocher, des boutons radios, des boutons à pousser, des menus à tiroirs, des étiquettes, des claviers, des souris et des stylos.

Cette idée de prendre une action simple ou un symbole et de l'imprégner de signification est bien connue des professionnels de la vente. Synthésiser des idiomes est l'essence même des marques: une entreprise choisit un nom de produit ou de société et l'imprègne de la signification souhaitée. Tylenol est un mot sans signification, un idiome, mais la société McNeil a dépensé des millions pour que les gens l'associent à un remède sûr, digne de confiance et efficace contre la douleur. Naturellement, les idiomes sont visuels, aussi. Les arches d'or de MacDonald's, les chevrons de Citroën, les cinq anneaux des Jeux Olympiques, et même les fenêtres «volantes» de Microsoft sont des idiomes non-metaphoriques qui sont immédiatement reconnaissables et imprégnés d'une signification commune.

Ironiquement, une grande partie des bagages familiers d'une interface graphique utilisateur (GUI) est souvent perçue comme metaphorique, alors qu'elle est réellement idiomatique. Des objets façonnés tels que les boîtes servant à fermer les fenêtres, les fenêtres redimensionnables, les dossiers imbriqués à l'infini ou encore le cliquer-déposer sont des opérations non-metaphoriques qui n'ont aucun parallèle dans le monde réel. Ils tirent leur force uniquement de leur apprentissage idiomatique facile.

**Le grain de sable dans l'engrenage[1].**

Si nous dépendons du fait de trouver des métaphores pour créer des interfaces-utilisateur, nous nous exposons donc aux différents problèmes mentionnés plus haut. Deux d'entre eux sont néanmoins plus importants: il est difficile de trouver des métaphores

1. Le terme initialement utilisé par Alan Cooper est «showstopper», mot utilisé dans le jargon informatique pour désigner un bug qui gèle complètement le travail de programmation.

et elles restreignent notre pensée.

Il peut être facile de découvrir des métaphores visuelles pour les objets physiques comme des imprimantes et des documents. Il peut être difficile voire impossible de trouver des métaphores correctes pour des processus, des rapports, des services ou des transformations, fonctions pourtant fréquentes des logiciels. Il peut être extrêmement délicat de trouver une métaphore visuelle utile pour acheter un billet, changer de chaîne, acheter un article, trouver une référence, choisir un format ou changer une résolution, et pourtant ces opérations sont précisément du type de celles que l'on trouve le plus fréquemment dans un environnement de travail.

Le problème le plus insidieux avec les métaphores, le grain de sable dans l'engrenage, survient lorsqu'on associe nos interfaces à des objets de l'âge mécanique. Il est facile d'apprendre intuitivement comment utiliser, par exemple, un presse-papier (clipboard), parce que c'est une métaphore. Mais après l'adhésion effective à cette métaphore, le service rendu se révèle incroyablement faible. Le presse-papier ne peut pas contenir plus d'une chose, il ne garde en mémoire aucun historique des manipulations, il ne permet pas d'identifier l'origine des images, aucune visualisation sous forme de vignette n'est disponible et il ne sauve aucun contenu d'une manipulation à l'autre. Toutes ces actions sont non-métaphoriques et devraient être néanmoins apprises. Suivre cette métaphore donne aux utilisateurs une impulsion lors de la première utilisation du presse-papier, mais elle masque considérablement et de manière durable la faiblesse arbitraire du service.

Autre exemple vraiment indigne: MagicCap[1], la nouvelle interface de communication produit par General Magic. Chaque aspect de son interface est exclusivement fondé sur des métaphores. Vous descendez metaphoriquement une rue longée de bâtiments représentant des services. pour commencer une tâche, vous entrez dans un bâtiment représenté par un vestibule garni de portes représentant des fonctions. Par le recours à cette métaphore, il est possible de comprendre intuitivement le fonctionnement de base du logiciel, mais le mauvais côté de cette approche est que la métaphore limite toute la navigation à un chemin très rudimentaire et linéaire. Si vous souhaitez effectuer une autre tâche, vous devez ressortir sur la rue pour trouver un autre service. Quoi de plus normal dans un monde physique, mais dans le monde du logiciel il n'y a aucune raison de forcer l'utilisateur à ces vieilles méthodes maladroites. Dans ce cas, pourquoi ne pas abandonnez cette dévotion à la métaphore qui nous en rend finalement esclave, et fournir à l'utilisateur des services accessibles sans devoir ressortir en rue ?

Qu'on ne me comprenne pas mal: il n'y a, à priori, pas de mal à utiliser une métaphore pertinente, lorsqu'elle se révèle adaptée à la situation. Si je vois traîner un billet de vingt dollars sur le trottoir, sûr que je le ramasse. Je serais bien bête de ne pas le faire ! Mais je serais encore plus bête d'en conclure que l'on peut vivre en trouvant des billets de vingt dollars en rue. Dans le cas des métaphores, c'est un peu la même chose: on les utilise lorsqu'on en trouve, mais pas la peine de forcer les interfaces à s'adapter à certaines normes métaphoriques arbitraires.

Il peut sembler pertinent de représenter un service d'appel par l'image d'un téléphone posé sur un bureau, mais cette approche vous emprisonne réellement dans une mauvaise voie. Les créateurs du téléphone auraient été fous de joie s'ils avaient pu en créer un qui vous aurait permis d'appeler vos amis en pointant simplement leur image. Cela ne leur a pas été possible parce qu'ils ont été limités par la morne réalité des circuits électriques et de du moulage du bakélite.

1. Logiciel, aujourd'hui disparu, conçus pour les PDAs Newton. Voir captures d'écran en annexes.

D’autre part, alors que nous avons aujourd’hui le luxe de rendre possible des communications en montrant des images de nos amis, pourquoi insister en incarnant ce type de situation par des images de technologies désuètes ?

La tentation d’étendre la métaphore au delà de la reconnaissance d’une fonction est irrésistible: ce petit téléphone de bureau vous permet également de «composer un numéro» avec des boutons en tout point identiques à ceux de votre propre téléphone de bureau. Nous constatons que le logiciel possède des «carnets d’adresses» de numéros de téléphone identiques à ceux que nous transportons dans nos poches ou sacoches. Ne vaudrait-il pas mieux dépasser ces technologies contraignantes et profiter de la vraie puissance de l’ordinateur ? Pourquoi nos dispositifs de communications ne permettraient-ils pas des connexions multiples, établies par organisation ou affiliation, pour finalement d’abandonner tout à fait ce système de numérotation téléphonique ?

Dans le domaine du design d’interface-utilisateur, le futur sera idiomatique, suite logique de la capacité normale des êtres humains à apprendre facilement et rapidement, tant que nous ne les forçons pas à comprendre le comment et le pourquoi. Il y a un infinité d’idiomes à inventer, plutôt qu’un éventail de métaphores à exploiter. Les métaphores semblent d’abord être un gain pour les utilisateurs débutants mais elles se montrent lourdes de conséquences lorsque l’on progresse dans l’utilisation approfondie d’un logiciel. Il est donc préférable de concevoir les choses de manière idiomatique, en utilisant la métaphore de manière occasionnelle, lorsque l’une d’elles nous tombe sous la main.

## Annexes.

Quelques exemples de l'interface du système d'exploitation MagicCap, aujourd'hui disparu, conçus pour les PDAs Newton.
Ces exemples, obsolètes, illustrent la réflexion d'Alan Cooper, toujours très actuelle dans la conception d'interfaces.


File Edit Hardware
11:22 PM
Hallway
Monday, June 17, 1996
Downtown
Directory
Game room
680x0.gif
ppo.gif
c64art.sit
uae.html
uae2.html
kick.rom
QL_ROM
OP_SYSTEM
Trash


File Edit Hardware
11:20 PM
Desk
Monday, June 17, 1996
Hallway
In: 1
Out: 1
'96
680x0.gif
ppo.gif
c64art.sit
uae.html
uae2.html
kick.rom
QL_ROM
OP_SYSTEM
Trash


File Edit Hardware
11:20 PM
Clock
Monday, June 17, 1996
Desk
June
S M T W T F S
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
1996
San Francisco
11:20 p.m.
display
world
set time
set date
680x0.gif
ppo.gif
c64art.sit
uae.html
uae2.html
kick.rom
QL_ROM
OP_SYSTEM
Trash