# ¿Estamos diseñando las interfaces de software correctas?

*Gerardo Cerda Neumann**

**Resumen**

*Actualmente, los computadores y el software que utilizan, están presentes en casi todas las actividades humanas: desde consultar un saldo bancario hasta subir en ascensor en un edificio "inteligente" pasando por la compra de bonos en un hospital y por el certificado de notas de un colegio. Sin embargo, a pesar de que el hardware es cada vez más poderoso y cada pocos meses se duplica su capacidad, no ocurre lo mismo con el software. Si bien ha mejorado la calidad y confiabilidad del mismo, existe un área que ha sido muy descuidada hasta el momento: el diseño de la interfaz del usuario. Lo más común es que se deje el diseño de la interfaz (si es que se hace) como última actividad en el desarrollo del software y, por lo general, se encomiende a personas que, aún siendo expertos usuarios de software, no tienen los conocimientos necesarios respecto a la mejor forma de que los usuarios finales (que no tienen por qué ser expertos ) saquen provecho a los programas. Si se revisa la malla curricular de las carreras que forman profesionales informáticos, se encontrará que en la gran mayoría existen pocas asignaturas que estudien el tema de la interfaz. Entonces se produce la extraña paradoja de que los diseñadores de software realizan esta actividad pensando en usuarios expertos y exigiendo a la mayoría de las personas que han de usar sus productos que se conviertan en uno más. Sin embargo ¿es justo eso? Claramente no. En este documento introductorio se revisarán algunos conceptos básicos que debe cumplir un buen diseño de interfaz y se harán algunas sugerencias de cómo aplicarlos.*



* Académico, Escuela de Ingeniería, UCINF

## 1. Introducción

A manera de introducción del tema se presentará lo que ha sido llamado "La prueba del pasillo del avión" (Cooper, 2001): para realizar esta prueba, sólo tiene que visualizarse caminando a lo largo del pasillo de un avión para abordar una nave. Al poner un pie en el avión, usted puede dar la vuelta a la izquierda, hacia la cabina de mando, o a la derecha, hacia la de pasajeros.

A la izquierda, la cabina de mando es un caleidoscopio de complejos controles y medidores; toda la superficie está cubierta de instrumentos, perillas y palancas. A la derecha, en agudo contraste, está la cabina de pasajeros, en donde todo tiene un contorno redondeado y suave y un plácido tono en beige.

Si da la vuelta a la derecha y se mete en la cabina de pasajeros, ello significa que cede toda autoridad sobre el vuelo. A cambio de ceder el control, se puede relajar, sabiendo que llegará al destino adecuado sin ocuparse de nada más complejo que prender o apagar la luz de lectura.

La idea es simple: el usuario quiere utilizar un software que sea lo más simple posible. No le interesa entender el por qué del funcionamiento sino solamente cómo lo debe utilizar para sacarle provecho. Sin embargo, si el diseño de la interfaz no es el adecuado, se estará obligando al usuario a aprender a utilizar complejos instrumentos para sacar provecho al software (imagínese que se obligara a los pasajeros de un vuelo a tener conocimientos aeronáuticos para poder abordar un avión).

A modo de ejemplo personal, se mencionará una experiencia ocurrida hace pocos días al realizar un giro de una cuenta de ahorro voluntario en una AFP. Como el pago se hace mediante un cheque, la sucursal posee dos impresoras: una de inyección de tinta para los certificados de renta y otra de matriz de punto para imprimir los cheques (que vienen en un formulario pre impreso en papel continuo). La persona que atiende estos giros tuvo que hacer los siguientes pasos:

1. Poner el papel continuo en la impresora de matriz de punto (no está siempre puesto por el temor de que alguien se equivoque y envíe a imprimir un certificado por esta impresora, echando a perder un cheque en blanco).

2. Ejecutar los pasos necesarios en el sistema operativo para llegar a la opción de cambio de impresora dado que normalmente el computador está imprimiendo en la de inyección de tinta.

3. Escoger la impresora de matriz de punto de la lista ofrecida cuidando de no equivocarse (afortunadamente sólo había dos opciones).

4. Volver al programa que estaba usando e imprimir el cheque.

5. Repetir los pasos 2 y 3, pero esta vez a la inversa para dejar seleccionada nuevamente la impresora de inyección de tinta.

6. Sacar el papel continuo de la impresora de matriz de punto.

¿Realmente eran necesarios tantos pasos? Obviamente no: hubiese bastado que el programa cambiara por sí mismo la impresora al ejecutar la opción de imprimir el cheque, pudiendo de esta forma tener siempre el papel continuo puesto.

¿La persona que programó no sabía hacer esto? Es casi seguro que si, pero como se puede usar el programa de la otra forma y evita escribir más códigos (en un proyecto de desarrollo probablemente atrasado) se escogió esa opción. Seguramente el programador hizo el fuerte compromiso de agregar la opción más adelante, pero ese "más adelante" nunca fue posible.

## 2. Características que debe cumplir una buena interfaz.

Se han realizado varias investigaciones en este tema y se ha llegado a ciertos acuerdos. A continuación se presentan algunas de las características deseables de una interfaz.

**Mantener sencillo lo sencillo:** la idea es que lo que es fácil de usar lo siga siendo y no se complique de manera artificial.

A modo de ejemplo se cita el ajuste de un reloj de video cassettera. ¿Quién no ha visto la hora pestañeando en uno de estos equipos? Se podrá argumentar que se debe a los cortes de electricidad, sin embargo, lo más probable es que el dueño no quiera poner la hora correcta debido a lo complicado que le resulta el ajuste. En este sentido se propone la siguiente interfaz (Raskin, 2001):


Lo más simple es tener cuatro botones: uno para aumentar las horas, otro para aumentar los minutos, uno para disminuir las horas y finalmente otro para disminuir los minutos.

¿Para qué complicar más las cosas?

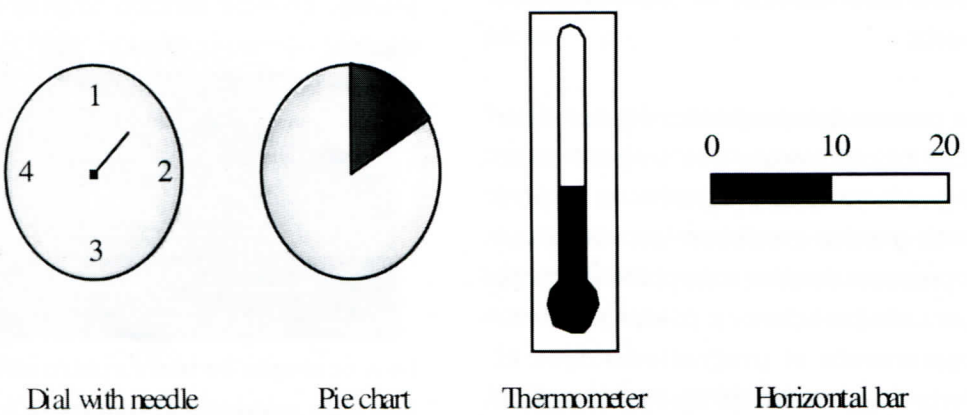
**Este es un principio importante:** mantener sencillo lo sencillo.

Es verdad. Las tareas complejas pueden requerir de interfaces complejas pero no hay motivo para complicar las tareas que son sencillas.

**Retroalimentar a los usuarios convenientemente:** se debe partir del supuesto que los usuarios se equivocarán (aún los más expertos). Por este motivo los mensajes del software deben ser lo más claros posibles (evitando la costumbre de poner la palabra ERROR que los hace sentir como estúpidos).

Una buena analogía se puede encontrar en la experiencia de manejar un automóvil. El conductor acelera, frena y mueve el volante sintiendo de manera instantánea que el automóvil responde. Como el software puede entregar sensaciones físicas aún de manera muy sencilla (y utilizando hardware complejo y caro) lo mejor es que el usuario tenga frente a sí claros indicadores de lo que está sucediendo. En este sentido los despliegues de información dinámica (que han sido creado imitando indicadores usados hace mucho tiempo por las personas) resultan muy útiles. A modo de ejemplo se pueden mostrar: (Ver figura 2)

(figura 2)



Diales tipo reloj, gráficos de torta,termómetros y barras horizontales, respectivamente. El objetivo es dar una información dinámica (cambiante) e instantánea. ¿Quién no se ha tranquilizado cuando ve en la interfaz del software una barra horizontal cuya "columna" va aumentando, aunque sea lentamente?

Además se puede mencionar el ejemplo de una búsqueda de información que no arrojó resultados positivos al no estar el paciente en la lista (Sommerville, 2002): (Ver figura 3)

El usuario del software encuentra esta ventana donde debe ingresar el nombre del paciente por el que se desea consultar. El mensaje que se incluye en la ventana es claro y directo, además del detalle de pedir por favor que se digite el nombre.

Si el nombre del paciente no está registrado, el software debe indicarlo de una manera clara y no ambigua. Aquí es cuando, generalmente, se comete el error de escribir un mensaje pensado para un usuario experto y no para un usuario común.

¿ Cuántas veces ha visto mensajes cómo el siguiente ? (Ver figura 4)



figura 3

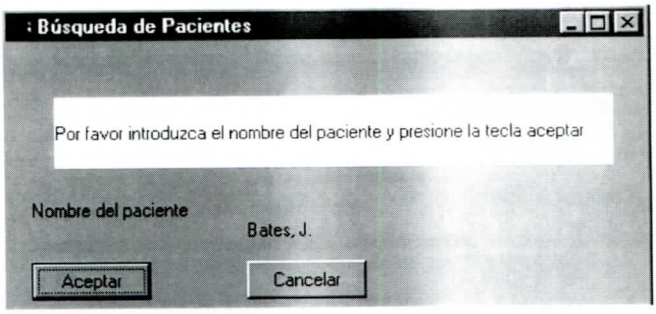
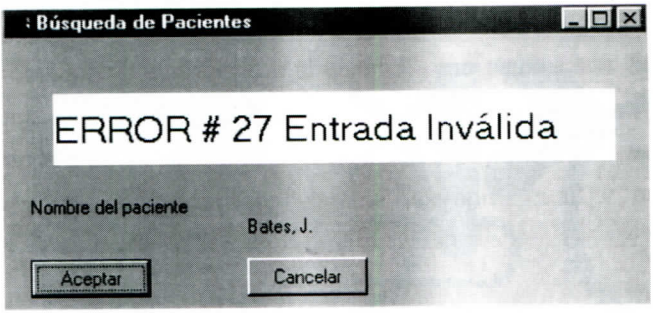



figura 4



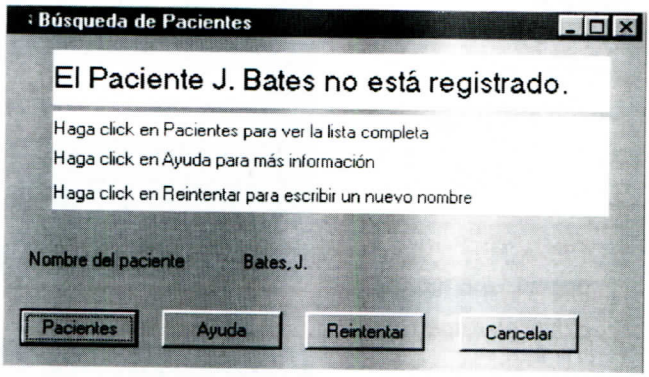
El programador creó una lista de mensajes que el software puede entregar y le asignó el número 27 cuándo se solicita algún dato que no se encuentra registrado. Sin embargo, a menos de que se disponga de la lista completa de estos mensajes, sólo incluir el número resulta incomprensible, o al menos confuso. Además el hecho de incluir la palabra error hace sentir al usuario como un inepto. ¿Es un error que un dato no se encuentre? Obviamente no. Si bien se podría considerar como error escribir mal el apellido indicárselo al usuario no disminuirá precisamente su incomodidad al usar la interfaz.

Sería mucho mejor utilizar la siguiente ventana: (Ver figura 5)

Se explica claramente que el paciente buscado no está registrado. Además se da la opción de: ver toda la lista de pacientes, pedir ayuda de uso, reintentar el ingreso del nombre del paciente buscado o simplemente salir de esta opción del software. Claramente esta interfaz representa un gran avance.

¿Cuántas veces ha visto algo así? Lamentablemente muy pocas.

figura 5



**Mantener un estilo definido:** cuando un software ha sido hecho por distintas personas el usuario se da fácilmente cuenta. Para la misma situación aparecen mensajes diferentes o a la inversa, para situaciones distintas se utiliza el mismo mensaje. En este sentido resulta muy importante mantener un estilo, sobre todo en lo gráfico, consistente. Esto implica, por una parte, usar tipos de letras, fondos y colores de manera consistente (siempre igual) y , por otra parte, lograr una combinación armoniosa.

A modo de ejemplo se presentará la siguiente situación (Kristof, 1998): se poseen varias imágenes relacionadas con el ciclismo y se necesita armar una ventana inicial de un club de ciclismo llamado "ACME".
Las imágenes que se desea utilizar son:

1 Una pintura de fines del siglo 19.

2 Una fotografía blanco y negro de estilo antiguo.

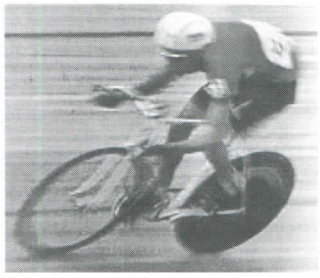
3 Una fotografía moderna de un ciclista de velocidad. Esta imagen es de gran dinamismo, en contraste con las dos anteriores.

4 Una imagen que representa el ícono del ciclismo.

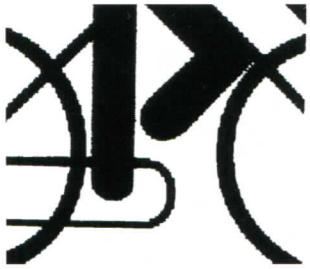


1

2



3



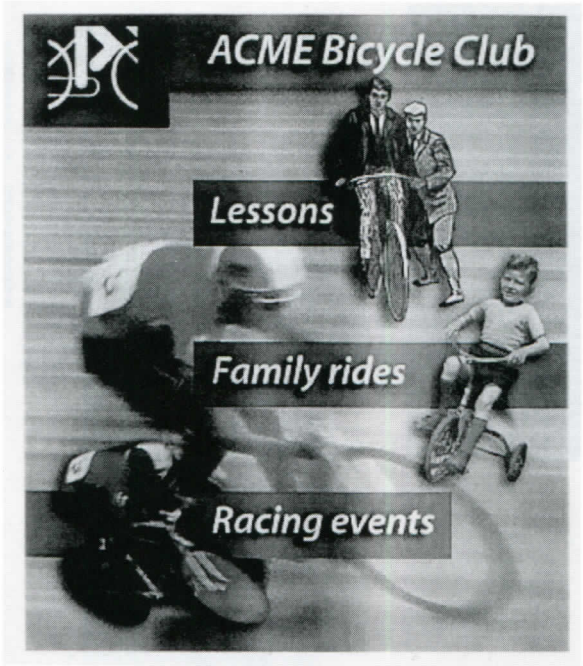
4

La duda es, ¿se puede combinar todas estas imágenes para lograr una buena interfaz sin caer en la incongruencia? La respuesta es la creación de la siguiente interfaz que presenta el club de ciclismo "ACME":(Ver figura 6)

Se le aplicó un efecto a cada imagen para aumentar la uniformidad y permitir la combinación y mezcla.El icono se usó como apoyo al nombre del club. El cuadro se usó para informar que hay lecciones de manejo de bicicletas (aprovechando la escena en que se aprecia claramente que se está enseñando a un alumno). La fotografía antigua se usó para destacar los paseos familiares. La imagen moderna se dio vuelta para dar la sensación de una mayor rapidez del ciclista (recuerde que al leer de izquierda a derecha es inevitable tener la sensación de una mayor rapidez cuando las imágenes parecen desplazarse en ese sentido).

El resultado final es armónico y útil. Se saca provecho a todos los elementos gráficos disponibles. Sin embargo, para su creación es necesario aplicar muchos conocimientos y experiencias en interfaces. El resultado bien lo merece.

figura 6



## 3. A modo de conclusión preliminar.

En este breve artículo se ha querido entregar algunos elementos para enriquecer el debate respecto a la interfaz.

Lo primero es que el éxito o fracaso de un software dependerá en gran medida de la calidad de la interfaz (Sommerville, 2002). Lo segundo es que es complejo diseñar una buena interfaz y que a esta actividad se le dedica, en general poco tiempo durante la construcción del software. Lo tercero es que no basta tener experiencia en el desarrollo de software, es necesario contar con la ayuda de un profesional calificado en el tema de la interfaz, de lo contrario se corre el riesgo de no cumplir las expectativas del usuario.

De hecho es común ver la interfaz de un software y saber con qué lenguaje de programación fue construído debido a que se utilizan todos los valores por defecto de la interfaz (tamaño, color y forma de los botones, tipos y tamaños de las letras, cajas de diálogo, etc.).

Por último, parece un tema demasiado importante como para dedicarle tan poco tiempo y esfuerzo en las mallas curriculares de las carreras informáticas.

El tema es apasionante y complejo, por lo que se espera haya aportes interesantes que permitan llegar al principal logro de una buena interfaz: hacer sentir al usuario del software tranquilo, feliz y capaz.

## Referencias.


1. Cooper,Alan. (2001). Presos de la tecnología. México, Prentice Hall, 259 p.

2. Kristof, Ray y Satran, Amy.(1998). Diseño interactivo. Madrid, Editorial Anaya Multimedia. 136 p.

3. Raskin, Jef. (2001).Diseño de sistemas interactivos. México, Addison Wesley, 272 p.

4. Sommerville, Ian.(2002). Ingeniería de Software. 6ª. Ed. México, Addison Wesley 692 p.